Proceedings of the 2010 Winter Simulation Conference
B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, eds.

TOWARDS AN ONTOLOGICAL FOUNDATION OF DISCRETE EVENT SIMULATION

Giancarlo Guizzardi

Computer Science Department
Federal University of Espírito Santo (UFES)
Av. Fernando Ferrari
s/n29060-970 Vitória, Espírito Santo, BRAZIL

Gerd Wagner

Institute of Informatics
Brandenburg University of Technology
P.O.Box 101344
03013 Cottbus, GERMANY

ABSTRACT

This paper is an attempt to transfer some results in the meta-theory of conceptual modeling of software systems to discrete event simulation modeling. We present DESO, a foundational ontology for discrete event system modeling derived from the foundational ontology UFO. The main purpose of DESO is to provide a basis for evaluating discrete event simulation languages.

1 INTRODUCTION

In recent years, there has been a growing interest in the application of *foundational ontologies* (also known as *upper level*, or *top-level* ontologies), i.e., formal ontological theories in the philosophical sense, for providing real-world semantics for conceptual modeling languages, and theoretically sound foundations and methodological guidelines for evaluating and improving the individual models produced 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 engineering in general, and of *Discrete Event Simulation* engineering in particular, did not yet receive much attention in the scientific literature.

A number of simulation researchers have investigated the use of ontologies and ontology engineering, mostly in connection with the Web Ontology Language OWL, for the purposes of simulation engineering (see Section 2). But, to our knowledge, there has been no attempt yet to demonstrate the value of an ontologically well-founded modeling language for simulation engineering. A simulation model is similar to a software system model in many respects. Following a scientific engineering approach, which seems to be more common in software engineering than in simulation engineering, both kinds of model are derived from a conceptual system model (also called *domain model*). In software engineering, this approach is called *Model-Driven Engineering*, following the *Model-Driven Architecture* proposal of the Object Management Group <www.omg.org/mda/>.

The main benefit obtained from establishing the ontological foundations of the core concepts of a conceptual modeling language is a clarification of its real world semantics. A clearly defined semantics of the conceptual model of a domain leads to a higher overall quality of the software application system built upon that domain model with respect to comprehensibility, maintainability, interoperability and evolvability. We may expect to obtain similar benefits in simulation engineering. In fact, simulation models are more about the real world than software system models, which are, to a large part, about implementation artifacts in addition to the entity-relationship representation of the real-world part they have to deal with. Therefore, using an ontologically well-founded modeling language seems to be even more relevant for simulation engineering than for software system engineering.

The notion of a simulation model subsumes many different kinds of formalisms and languages, from more abstract approaches, such as *Classical Petri Nets*, *System Dynamics* and *DEVS* (Zeigler 1976), to programming frameworks such as *RePast* `<repast.sourceforge.net>`. There is widespread agreement that the paradigm of *Discrete Event Simulation* (DES) forms a core discipline of simulation. However, there is no commonly agreed-upon precise definition of DES and its underlying concepts, such as *entity*, *object*, *event* and *state*. In this article, we focus on DES, as we think that it is the most fundamental simulation paradigm.

The entire field of computer simulation is suffering from a plethora of different concepts, formalisms and technologies and from a lack of common modeling languages and standards. While in the field of software engineering, the *Unified Modeling Language (UML)* and the *Business Process Modeling Notation (BPMN)* have been established as widely used modeling language standards based on a large body of scientific work, there are no equivalents in the field of simulation engineering. We believe that both UML and BPMN, with suitable modifications and enhancements based on their ontological analysis and improvement, have a high potential as simulation modeling languages.

In a series of publications (Guizzardi and Wagner 2004+2005+2010; Guizzardi, Falbo and Guizzardi 2008) 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. This work started with (Guizzardi, Herre and Wagner 2003) as an effort to unify the *Generalized Formal Ontology* (Heller and Herre 2004) and the top-level ontology of universals from *Onto-Clean* `<http://www.ontoclean.org>`. The core of UFO has been established through the development of an ontology of endurants by the first author in (Guizzardi 2005). This foundational ontology has been successfully applied in the analysis of several important conceptual modeling constructs such as Roles, Types, Part-Whole Relations, Attributes, Datatypes, among others.

In this article, using the UFO layers A and B (about objects and events), we first analyze the ontological foundations of basic DES. In a follow-up article, using the UFO layers C and D (about simple and cognitive agents) we discuss the ontological foundations of multi-agent systems and of *agent-based* DES.

The remaining of this article is organized as follows. In section 2, we discuss related work. In section 3, we present a simplified version of UFO, tailored to the purposes of this article.

2 RELATED WORK

As a consequence of the rising interest in ontologies for the Semantic Web beginning around the year 2000, also simulation researchers working in different areas started to investigate the use of ontologies for simulation. In (Fishwick and Miller 2004), the authors report about two different efforts involving ontologies: a) the RUBE project aims at providing an XML-based simulation modeling framework supporting both 2D and 3D models, and b) the DeMO project aims at establishing an ontology for discrete event simulation. As explained in (Silver et al. 2009), the main concern of DeMO is to support *ontology-driven simulation*. The starting point for the DeMO methodology is the conceptual model of a system obtained as the first step in the process of making a simulation model. This model has to be provided in the form of an OWL ontology. It is then mapped to an instantiation of the DeMO ontology, which is, in turn, mapped to an executable simulation model. Thus, the DeMO ontology constitutes a high-level simulation language supporting the paradigms of ‘state-oriented’, ‘event-oriented’, ‘activity-oriented’ and ‘process-oriented’ simulation. The conceptual framework proposed by DeMO is, however, not based on a foundational ontology.

In (Christley, Xiang and Madey 2004), an OWL ontology defining an agent-based simulation framework is presented and possibilities for using OWL’s automated reasoning capabilities are discussed. How to benefit from OWL and Semantic Web technologies for simulation is also the topic of (Pignotti et al. 2005), where the goal is to enhance a social simulation tool.

In (Benjamin, Patki and Mayer 2006), it is recommended to use domain ontologies in the simulation modeling process for making simulation models unambiguous and consistent. It is argued that in distributed simulation, ontologies may play the role of a vendor/platform-independent modeling language that facilitates the translation of models into the different simulation platforms involved in a distributed simulation.

In (Livet et al. 2010), 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.

So, while there have been several proposals about how to use (OWL) ontologies in simulation engineering, we were not able to find any work on the ontological foundations of simulation (modeling) languages.

3 INTRODUCTION TO THE ESSENTIAL UFO

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.

The base layer of eUFO, called eUFO-0, is depicted in Figure 1 below. UFO makes a fundamental distinction between the categories of ***Individual*** and ***Universal***. Individuals are things that exist in space and time in "the real world" and have a unique identity, while *universals* are feature-based classifiers, which classify, at an moment in time a number of different individuals. An example of an individual is *the earth*; an example of a universal is the entity type *planet*.

We distinguish between three kinds of individuals: *substance individuals* (also called *endurants*), *trope individuals* and *events* (also called *perdurants*). 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 time*. Events *happen in time*, they may have temporal parts.

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, the sinking of the Titanic, the Second World War, or a particular business process. 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 further discussed in subsection 3.1, while the ontology of events forms the UFO layer B, which is further discussed in subsection 3.4.

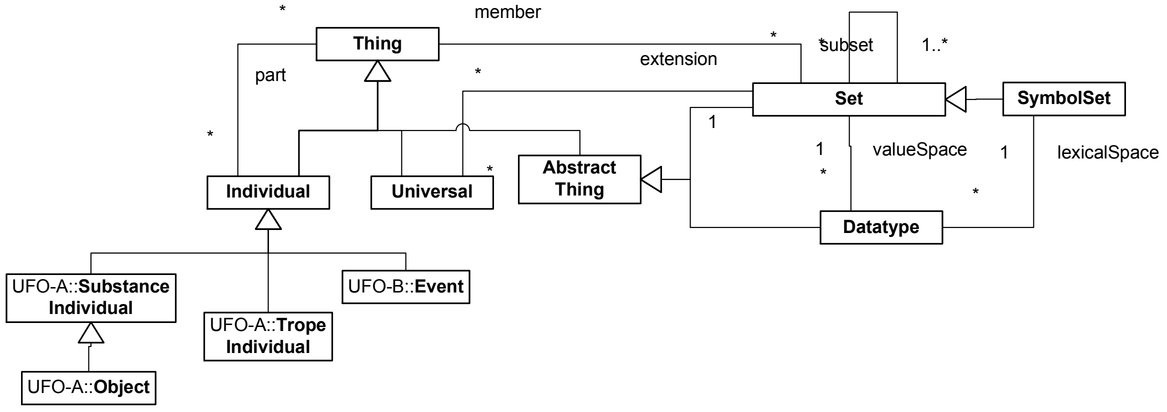

Figure 1: eUFO-0, the base layer of eUFO

In addition to individuals and universals, there are also ***abstract things***, such as *sets* and *datatypes*. In English, the term 'abstract' is overloaded, and one may consider the universal *planet* to be 'more abstract' than the individual *earth*. However, we consider things to be *abstract*, if they do not depend in any way on time and space.

Datatypes are an important concept in programming languages and in information modeling. We adopt the basic datatype concept proposed in [RDF Semantics], where a datatype is a triple consisting of a *lexical space* (a symbol set), a *value space* and a mapping from symbols to values (an extended concept of datatypes would also consider datatype-specific functions and predicates, that is datatypes as algebras).

Anything can be a member of a set. The *empty set* has no members and is a subset of all sets. The empty set is a *pure set*. Any set that has only pure sets as members is a *pure set* (in set theory, those things that are not sets are called 'urelements'). The set of all instances of a universal forms its *extension*.

A thing may be a *part* of an individual. There is a large body of work on the semantics of the part-whole relationship, but we do not discuss this topic here.

3.1 Substance Individuals

Substance individuals possess (direct) spatio-temporal qualities and are founded on matter. We distinguish between two kinds of substance individuals: *amounts of matter* and *physical objects* (however, the former are not of much interest to information modelling and DES). Examples of physical objects, which we also call just "objects", include ordinary entities of everyday experience such as my car, Alan Turing, The Rolling Stones, or the North-Sea. In contrast with trope individuals, substance individuals do not inhere in anything and, as a consequence, they are (essentially) *existentially independent*.

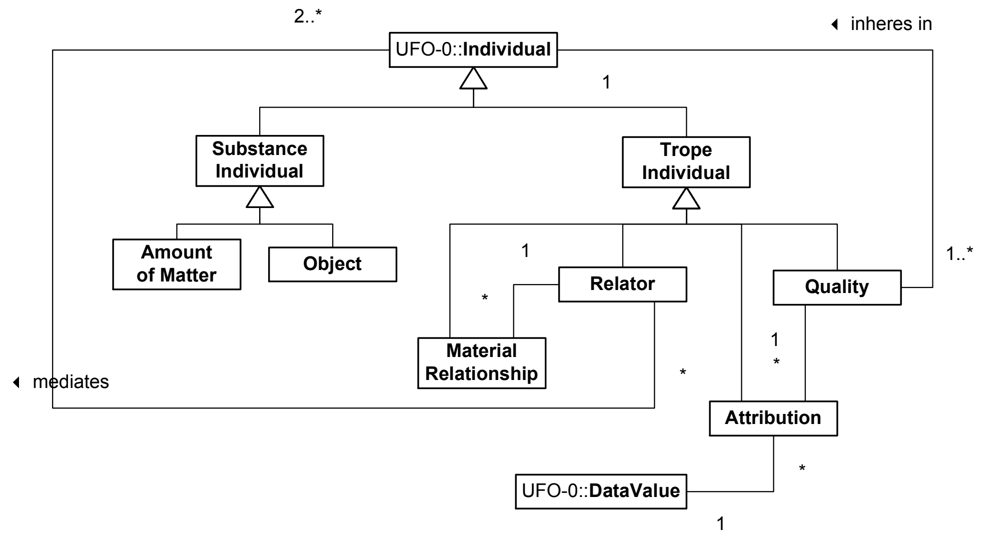

Figure 2: eUFO-A

3.2 Trope Individuals

The redness of John's T-shirt is an example of a trope individual. It *inheres* in John's T-shirt, which is its bearer. Both John's T-shirt and the redness of John's T-shirt are individuals. However, they are individuals of very different natures. Trope individuals can only exist in other individuals, i.e., they are *existentially dependent* on other individuals in the way, for instance, the color and the weight of an apple *a* depend on *a*, the electric charge of a conductor *c* depends on *c*, or John's headache depends on John. In contrast, individuals such as John, the apple *a*, and the conductor *c* are not existentially dependent entities in this sense.

There are two kinds of trope individuals: (a) *Intrinsic* trope individuals or ***qualities*** such as an individualized color, temperature, or weight, a symptom, a skill, a belief, an intention, an electric charge; (b)

Relational trope individuals or ***relators***: a kiss, a covalent bond, a medical treatment, a purchase order, or a social commitment.

The special type of existential dependence relation that holds between a trope individual x and the individual y on which x depends is the relation of *inherence*. Existential dependence can also be used to differentiate intrinsic and relational trope individuals: qualities are dependent on one single individual; relators depend on two or more individuals (their *bearers*), which they *mediate*.

3.2.1 Qualities and Attributions

We adopt the *non-migration* (or *non-transferability*) *principle*, which says that it is not possible for a particular quality *q* to inhere in two different individuals *a* and *b*. This principle may seem counterintuitive. For example, if we have two particulars *a* (a red apple) and *b* (a red car), and two qualities r_1 (the particular redness of *a*) and r_2 (the particular redness of *b),* we consider r_1 and r_2 to be different individuals.

What does it mean then to say that *a* and *b* have the same color? Sameness here cannot refer to strict (numerical) identity, but only to a qualitative one (i.e., equivalence in a certain respect). We thus distinguish between the color of a particular apple (its quality) 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 associated with the corresponding quality universal (see Subsection 3.5).

In general, we may associate more than one attribute with a particular quality universal, 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. Each attribution represents a *fact* that may be a "truth maker" for a corresponding sentence expressed in a suitable language.

3.2.2 Relators, Material Relationships and Reference Properties

While a ***formal relationship*** (such as $1 < 2$) 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.

For any relator that mediates certain entities, there is a corresponding *material relationship* based on a tuple constituted by these entities. A ***reference property*** is a binary material relationship.

In a correspondence theory of truth (such as Tarski's semantics of predicate logic), material relationships, and the attributions discussed above, can be viewed as "truth makers" ("facts") that make corresponding sentences true.

An important notion for the characterization of relators (and, hence, for the characterization of material relationships) is the notion of *foundation*, which can be viewed as a type of historical dependence in the way that, for example, an instance of being kissed is founded on an individual kiss. Suppose that John is married to Mary. In this case, we can assume that there is a particular relator m_1 of type marriage that mediates John and Mary. The foundation of this relator can be a wedding event or the signing of a contract between the involved parties. In other words, a certain event e_1, in which John and Mary participate, can create a particular marriage m_1 which existentially depends on John and Mary and which mediates them. The event e_1, in this case, is the foundation of the relator m_1. In general, relators are founded on events.

3.3 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. Examples of events are: the arrival of an incoming email message, a football game, a symphony execution, a birth of a mammal, the Second World War, or a particular business process. An event cannot exhibit change in time in a genuine sense since none of its temporal parts retain their identity through time.

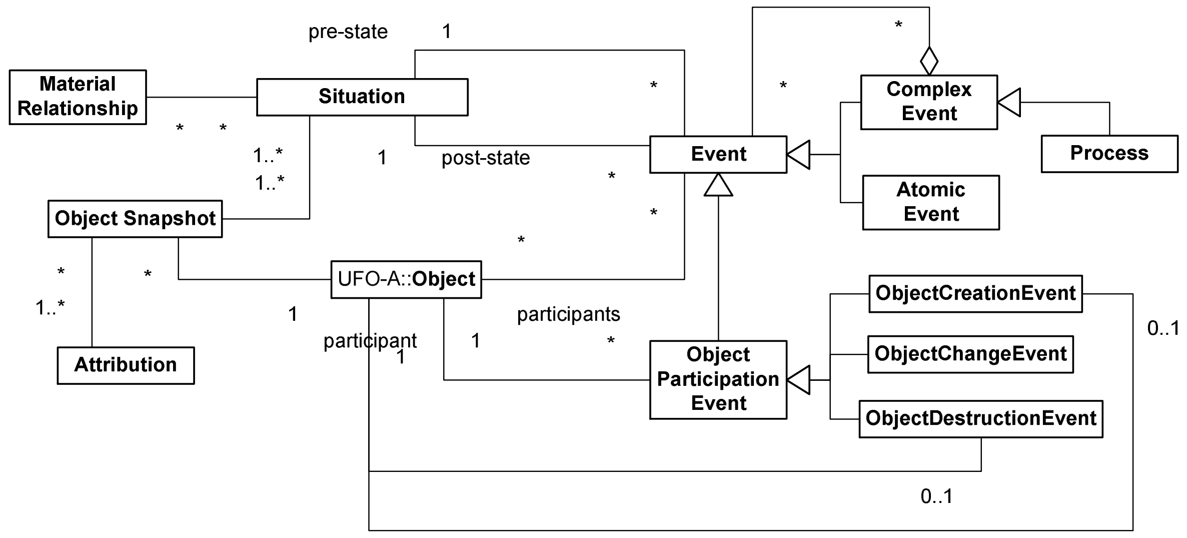

Figure 3: eUFO-B, an ontology of events

Figure 3 depicts the core fragment of the eUFO-B ontology of events. The main category of this ontology is *Event*. An event can be atomic or complex, depending on its mereological structure, i.e., while an *atomic event* has no parts, a *complex event* is an aggregation of at least two events (that can themselves be atomic or complex).

Events are ontologically dependent entities in the sense that they existentially depend on their *participants* in order to exist. Take for instance the event *e*: the stabbing of Caesar by Brutus. In this event 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 one (*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 about a particular object.

Being atomic and being instantaneous are orthogonal notions in this framework, i.e., an atomic event can be time-extended and an instantaneous event can be composed of multiple (instantaneous) events.

All spatial properties of events are defined in terms of the spatial properties of their participants. In contrast, all temporal properties of objects are defined in terms of the events in which they participate. The temporal attributes of events have values from special temporal datatypes. We assume that these datatypes support the concept of *Time Intervals*, which are composed of *Time Points*. Time points could be represented as real numbers and Time Intervals as sets of real numbers. However, they could also be defined in other ways (we avoid making unnecessary ontological commitments at this point). Additionally, we admit: (i) intervals that are delimited by begin and end points as well as open intervals; (ii) continuous and non-continuous intervals; (iii) intervals with and without duration (instants). In particular, this model allows a diversity of temporal structures such as linear, branching, parallel and circular time.

Finally, we would like to point out that a *process*, such as a chemical process or a business process, is a complex event.

3.4 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 five 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 five are the most relevant for conceptual modeling.

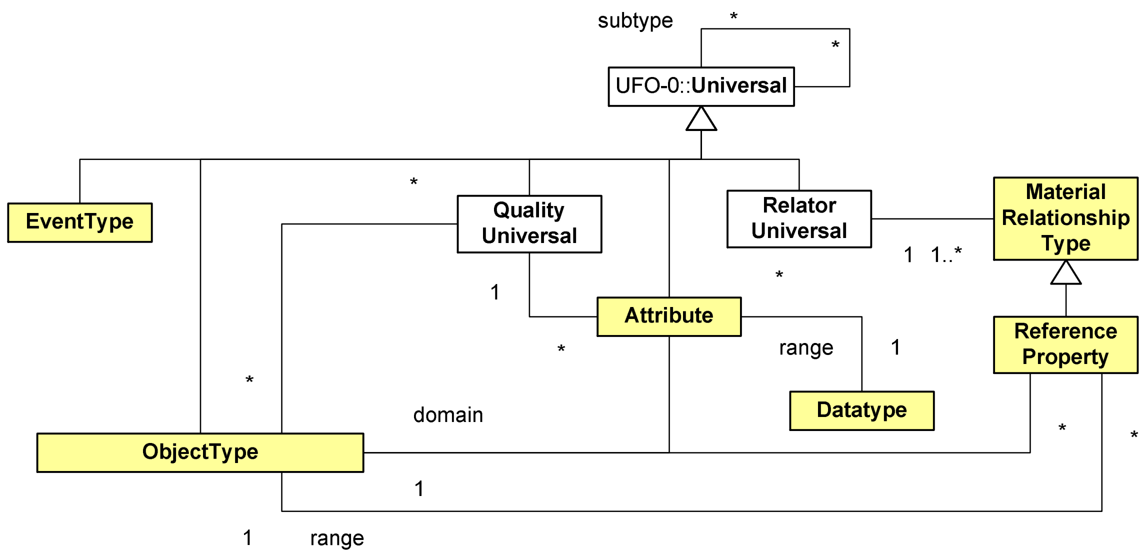

Figure 4: eUFO-U, an ontology of universals

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.

A universal is a *subtype* of another universal if its extension is a subset of the extension of the other universal. Unlike datatypes and sets, universals do not have a standard set-theoretical *extensional semantics*. That is, if O_1 and O_2 denote two object types that happen to have the same extension, they still do not denote the same universal.

3.4.1 Different Kinds of Object Types

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

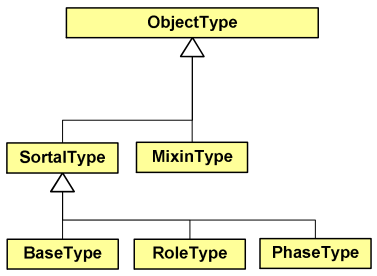

Figure 5: The different kinds of object types of eUFO-U.

3.4.1.1 Sortal Types and Mixin Types

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. As an illustration of this point, contrast the two object types Apple and RedThing instantiated by two individuals *x* and *y*: both object types supply a principle according to which we can judge whether *x* and *y* are classified under those types. However, only the object typeApple supplies a principle which allows to decide whether *x* and *y* are the same (i.e., merely knowing that *x* and *y* are both red gives no clue to decide whether or not *x*=*y*). Non-sortal types, such as RedThing, are called *mixin types*.

3.4.1.2 Base 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*.

An example that highlights this distinction is the difference between the base type Person and the anti-rigid object types Student and Adolescent instantiated by the individual John in a given circumstance. While John can cease to be a Student and Adolescent, he cannot cease to be a Person. In other words, while the instantiation of the object types Student and Adolescent has no impact on the identity of an individual, if an individual ceases to instantiate the base type Person, then she ceases to exist as the same individual.

3.4.1.3 Role Types and Phase Types

John can move in and out of the object type Student, while being the same individual, i.e. without losing his identity. This is because the principle of identity that applies to instances of Student and, in particular, that can be applied to John, is the one which is supplied by the base type Person of which Student is a subtype. 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: *role types* and *phase types*. Phase types constitute possible stages in the history of an individual. Examples include: (a) Alive and Deceased: as possible stages of a Person; (b) Catterpillar and Butterfly of a Lepidopteran; (c) Town and Metropolis of a City; (d) Boy, Male Teenager and Adult Male of a Male Person.

Role types differ from phase types with respect to the specialization condition SC. For a phase type P, SC represents a condition that involves only intrinsic properties of P. For instance, one might say that if John is a Living Person then he is a Person who has the property of being alive or, if Spot is a Puppy then it is a Dog that has the property of being less than one year old. For a role type R, conversely, SC involves extrinsic (relational) properties of R. For example, one might say that if John is a Student then John is a Person who is enrolled in some educational institution (playing the role of a student), if Peter is a Customer then Peter is a Person who buys a Product x from a Supplier y (playing the role of a customer), or if Mary is a Patient than she is a Person who is treated in a certain medical unit (playing the role of a patient).

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

In (Guizzardi 2005), it is formally proven that the following constraints hold:

1. Every object must instantiate exactly one ultimate base type.
2. A rigid object type cannot be a subtype of an anti-rigid object type (e.g., Person cannot specialize Student).
3. A mixin type cannot be specialized by a sortal type (e.g., Person cannot specialize Customer).
4. A mixin type cannot have direct instances.

3.4.2 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 of some quality universal and the corresponding data value from an associated datatype is provided by an ***attribute***. A quality universal can be captured by one or more corresponding attributes, each of them based on a different datatype. An attribute is a universal that classifies ***attributions***, which are 'truth makers' for attribution statements.

3.4.3 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 objects. Its tuple extension is a subset of the Cartesian product of the extensions of the two involved object types. The first object type is called the ***domain***, and the second one the ***range*** of the reference property.

4 A FOUNDATIONAL ONTOLOGY FOR BASIC DES

Basic DES is concerned with the simulation of real-world systems that are conceived as discrete event systems (or 'discrete dynamic systems'). Such discrete system *abstractions* are immaterial entities that only exist in the mind of the user (or a community of users) of a language. In order to be documented, communicated and analyzed they must be captured, i.e. represented in terms of some concrete artefact with the help of a language. The representation of an abstraction in a language is called a *model* and the language used for its creation is called a *modeling language*.

A discrete event system model may be expressed at different levels of abstraction. The system's state structure can be described by means of a set of variables and/or objects of some type. For the purpose of DES, there is a requirement that the (simulation) model must be executable by simulating the successive occurrence of events that may change the value of variables and the state of objects as well as create follow-up events.

The standard view in the simulation literature (see, e.g., Himmelspach 2009) is that a 'simulation model' can be expressed either in a general purpose programming language or in a specialized simulation language. This means, that the term 'model' is used rather loosely both for low-level computer programs and for higher-level executable specifications. There is often no distinction between a conceptual/logical system model (expressed either as a non-executable conceptual model or as an executable specification in a high-level simulation language) and its implementation in some target technology platform. Clearly, as in software engineering, such a distinction would be important for several reasons: as opposed to a low-level computer program, a high-level simulation model would be more comprehensible and easier to communicate, share, reuse, maintain and evolve, while it could still be transformed into any platform-specific implementation code.

In the DES literature, it is often stated that DES is based on the concept of "entities flowing through the system"; e.g., this is the paradigm of simulation software such as ARENA `<www.arenasimulation.com>`. However, the loose metaphor of a "flow" applies to the flow of events, to the flow of messages (in communication) and to the material flow of physical objects when they are processed in a manufacturing system or transported in a logistics system. Subsuming all these different kinds of flows under the term "entity flow" leads to a non-lucid simulation language (as we discuss below in Section 5). Clearly, DES is based on the concept of an *event flow* (a sequence of events occurring at times from a discrete set of time points), rather than on the concept of an "entity flow".

On the basis of eUFO, we will now construct a foundational DES ontology, called "DESO", as a representation of our conceptualization of discrete event systems. DESO will not contain all the foundational concepts of UFO, but rather just those leaf nodes of UFO's concept hierarchy, which are relevant for describing discrete event systems.

4.1 The Run-Time Ontology DESO-I

At run-time, a discrete event simulator deals with individuals of different types. From eUFO-A (see Figure 2), we just need the concept of *objects*. However, for pragmatic reasons, we rename it to *physical objects*, for being able to use the generic term 'object' in new sense: as a kind of wildcard that we can use for all kinds of not necessarily physical 'objects', including 'objectified' material relationships (such as, for instance, a book, being a material relationship between an author and a publisher).

From eUFO-B (see Figure 3), we need the concepts of *situations* and *object snapshots*, as well as most of its event concepts, as depicted in Figure 6.

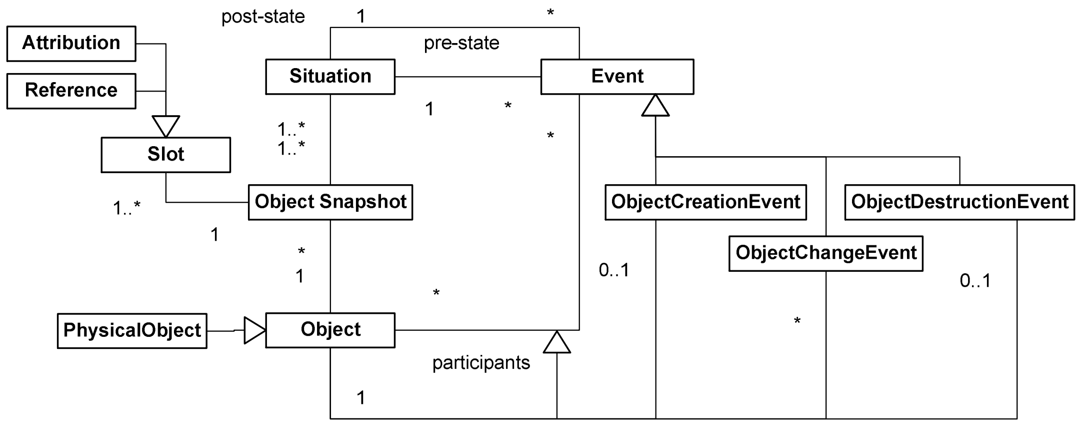

Figure 6: The categories of individuals, as defined by DESO-I

In DESO-I, for simplicity, we assume that there are only binary material relationships, which are represented by ***references*** specifying a value for a reference property. A ***slot*** is either a reference or an attribution (an attribute-value pair).

Notice that in conceptual modelling, and in simulation modelling, 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 amounts of matter, relators or qualities in a simulation model.

All these kinds of entities are classified with the help of *entity types*, as explained below in Subsection 4.2. 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.

4.2 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 system. In addition to the base concepts from eUFO-0 (see Figure 1), also the concepts of (physical) object types, attributes and reference properties from eUFO-U (see Figure 4) 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* (based on corresponding quality universals) and *reference properties* (based on corresponding material relationship types), as depicted in Figure 7.

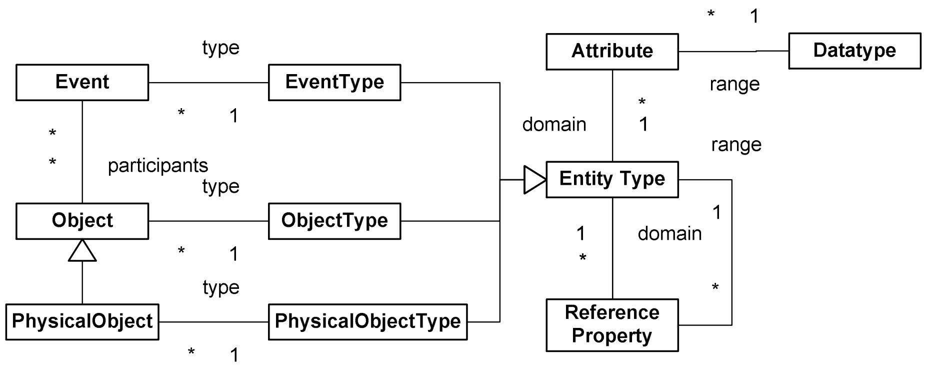

Figure 7: The categories of universals, as defined by DESO-U

4.3 Evaluating DES Languages

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

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

There are four properties of a simulation language to be checked in its evaluation:

1. ***Soundness***: *L* is sound w.r.t. DESO iff every element of *L* has an interpretation in terms of a domain concept from DESO. The ***degree of soundness*** can be measured relatively as the number of *L* elements that have a DESO interpretation divided by the total number of *L* elements.
2. ***Completeness***: *L* is complete w.r.t. DESO iff every DESO concept is represented by a modeling primitive of *L*. The ***degree of completeness*** can be measured relatively as the number of DESO concepts that are represented by an element of *L* divided by the total number of DESO concepts.
3. ***Lucidity***: *L* is lucid w.r.t. DESO iff every element of *L* has at most one interpretation in DESO. The ***degree of lucidity*** can be measured relatively as the number of *L* elements that have at most one interpretation in DESO divided by the total number of *L* elements.

4. ***Laconicity***: *L* is laconic w.r.t. DESO iff every domain concept from DESO is represented by at most one element of L. The ***degree of laconicity*** can be measured relatively as the number of DESO concepts that are represented by at most one element of *L*.

The lower these degrees are for a given simulation 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.

5 CONCLUSIONS

We have defined the *Discrete Event Simulation Ontology (DESO)*, derived from the *Unified Foundational Ontology (UFO)*. DESO defines the basic concepts that need to be supported in any general-purpose DES language. In future work, we plan to evaluate a number of established DES frameworks (such as Petri nets and DEVS) by analyzing the representation mapping from DESO to these languages, and the interpretation mapping from these languages to DESO.

REFERENCES

Benjamin P., M. Patki, and R. Mayer. 2006. Using Ontologies for Simulation Modeling. In *Proceedings of the 2006 Winter Simulation Conference*, eds. L. R. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, 1151–1159. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Christley S., X. Xiang, and G. Madey. 2004. An Ontology for Agent-Based Modeling and Simulation. Proc. of the Agent 2004 Conference.

Fishwick, P.A., and J. A. Miller. 2004. Ontologies For Modeling And Simulation: Issues and Approaches. In *Proceedings of the 2004 Winter Simulation Conference*, eds. R. G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, 259-264. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Guizzardi, G. 2005. Ontological Foundations for Structural Conceptual Models, PhD Thesis, University of Twente, The Netherlands.

Guizzardi, G., R.A. Falbo, and R.S.S. Guizzardi. 2008. 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), 2008, Recife.

Guizzardi, G., H. Herre, and G. Wagner. 2003. On the General Ontological Foundations of Conceptual Modeling. In Proceedings of Conceptual Modeling – ER 2002, Tampere, Finland, October 7–11, 2002. Springer Berlin/Heidelberg, Lecture Notes in Computer Science, Vol. 2503, pp. 65–78.

Guizzardi, G., and G. Wagner. 2004 . A Unified Foundational Ontology and some Applications of it in Business Modeling. In: Janis Grundspenkis and Marite Kirikova (Eds.), Proceedings of the CAiSE'04 Workshops. Faculty of Computer Science and Information Technology, Riga Technical University, Riga, Latvia. June 7–11, 2004. Volume 3. pp. 129–143. Riga, Latvia.

Guizzardi, G., and G. Wagner. 2005. 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, volume 3508, pp. 110–124, Springer Berlin/Heidelberg.

Guizzardi, G., and G. Wagner. 2010. 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.

Heller, B., and H. Herre. 2004. Ontological Categories in GOL. *Axiomathes* 14: 71–90.

Himmelspach, J. 2009. Toward a Collection of Principles, Techniques and Elements of Modeling and Simulation Software. Proc. of the 2009 International Conference on Advances in System Simulation. IEEE Computer Society, 2009, pp. 56–61.

Livet, P., J.-P. Müller, D. Phan, and L. Sanders. 2010. Ontology, a Mediator for Agent-Based Modeling in Social Science. *Journal of Artificial Societies and Social Simulation* **13**:1, Available via <http://jasss.soc.surrey.ac.uk/13/1/3.html> [accessed April 9, 2010].

Pignotti, E., P. Edwards, A. Preece, G. Polhill, and N. Gotts. 2005. Providing Ontology Support for Social Simulation. *Proceedings of First International Conference on eSocial Science*, pp. 22–24.

Rosemann, M., J. Recker, M. Indulska, and P. Green. 2006. A Study of the Evolution of the Representational Capabilities of Process Modeling Grammars. In E. Dubois and K. Pohl (eds.), *Advanced Information Systems Engineering*, Proceedings 18th International Conference, CAiSE 2006, Springer, LNCS Vol. 4001, pp. 447-461.

Silver, G. A., K. R. Bellipady, J. A. Miller, W. S. York, and K. J. Kochut. 2009. Supporting Interoperability Using the Discrete-Event Modeling Ontology (DeMO). In *Proceedings of the 2009 Winter Simulation Conference*, eds. M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, 1399–1410. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Zeigler, B. 1976. *Theory of Modeling and Simulation*. Wiley Interscience, New York.

AUTHOR BIOGRAPHIES

GIANCARLO GUIZZARDI is an Associate Professor at the Computer Science Department, Federal University of Espírito Santo (UFES), Brazil. His research is focused on the application of foundational ontologies in the development of philosophically and cognitively well-justified methodological tools for conceptual modeling in computer science. His email address is <gguizzardi@inf.ufes.br>.

GERD WAGNER is Professor of Internet Technology within the Department of Informatics, Brandenburg University of Technology. His research interests include agent-oriented modeling and agent-based simulation, foundational ontologies, (business) rule technologies and the Semantic Web. In recent years, he has been focusing his research on the development of an agent-based discrete event simulation framework, called *AOR Simulation*. He can be reached at <http://www.informatik.tu-cottbus.de/~gwagner/>.