Multi-Level Ontology-based Conceptual Modeling

Victorio A. Carvalho[1,2], João Paulo A. Almeida[1],
Claudenir M. Fonseca[1] and Giancarlo Guizzardi[1,3]

[1]Ontology & Conceptual Modeling Research Group (NEMO)
Federal University of Espírito Santo (UFES), Vitória, ES, Brazil
[2]Research Group in Applied Informatics, Informatics Department,
Federal Institute of Espírito Santo (IFES), Colatina, ES, Brazil
[3]Faculty of Computer Science, Free University of Bozen-Bolzano, Italy
victorio@ifes.edu.br; jpalmeida@ieee.org; claudenirmf@gmail.com; gguizzardi@inf.ufes.br

Abstract. Since the late 1980s, there has been a growing interest in the use of foundational ontologies to provide a sound theoretical basis for the discipline of conceptual modeling. This has led to the development of ontology-based conceptual modeling techniques whose modeling primitives reflect the conceptual categories defined in a foundational ontology. The ontology-based conceptual modeling language OntoUML, for example, incorporates the distinctions underlying the taxonomy of types in the Unified Foundational Ontology (UFO) (e.g., kinds, phases, roles, mixins, etc.). This approach has focused so far on the support to types whose *instances are individuals* in the subject domain, with no provision for *types of types* (or categories of categories). In this paper we address this limitation by extending the Unified Foundational Ontology with the MLT multi-level theory. The UFO-MLT combination serves as a foundation for conceptual models that can benefit from the ontological distinctions of UFO as well as MLT's basic concepts and patterns for multi-level modeling. We discuss the impact of the extended foundation to multi-level conceptual modeling.

Keywords: Ontology, Conceptual Modeling, Multi-level Modeling.

1 Introduction

Conceptual modeling is the activity of formally describing some aspects of the physical and social world around us for the purposes of understanding and communication [1]. It is generally considered a fundamental activity in information systems engineering [2], in which a given subject domain is described independently of specific implementation choices [3]. The main artifact of this activity is a conceptual model, i.e., a specification aiming at representing a conceptualization of the subject domain of interest.

Since the late 1980s, there has been a growing interest in the use of foundational ontologies to provide a sound theoretical basis for the discipline of conceptual modeling [4, 5, 6]. The initial hypothesis, which was later confirmed by different empirical evidence can be explained by the following arguments: (i) conceptual models are artifacts produced with the aim of representing a certain portion of reality according to a specific conceptualization; (ii) foundational ontologies describe the categories that are used for the development of these conceptualizations. Therefore, an appropriate conceptual modeling language should provide modeling primitives that reflect the conceptual categories defined in a foundational ontology. This observation has led to the development of approaches for conceptual modeling based on foundational ontologies. An example of such an approach is OntoUML, which is based on the Unified Foundational Ontology (UFO) [3].

In OntoUML, the taxonomy of types of the Unified Foundational Ontology (UFO) has been reflected in the language such that the distinctions of the foundational ontology can be used to provide useful constraints and modeling guidelines, ultimately leading to ontologically well-founded conceptual models. The resulting conceptual models consist of a collection of *types (classes) of individuals in the subject domain* (e.g., the "Person" kind, the "Child" phase, the "Student" role). Each of these domain types instantiate types in the foundational ontology (e.g., kind, subkind, role, phase, etc.).

The approach is so far unable to describe subject domains in which the categorization scheme itself is part of the subject matter. In these subject domains, experts make use of categories of categories in their accounts. For instance, considering the domain of human resource management, organizations are often staffed according to *employee types* (e.g. "Engineer", "Pilot", "Secretary"). They may need

to classify those *employee types* giving rise to *types of employee types*. In this case, "Engineer" and "Pilot" could be considered as examples of "Technical Employee Type", as opposed to "Secretary" which is an example of "Administrative Employee Type". Finally, they need to track the allocation of personnel to specific departments (e.g., to capture the fact that John is an Engineer in the Maintenance Department). Thus, to describe the conceptualization underlying this domain, one needs to represent entities of different (but nonetheless related) classification levels, such as *individual persons* ("John"), *employee types* ("Engineer", "Pilot", "Secretary"), and *types of employee types* ("Technical Employee Type", "Administrative Employee Type").

The need to support the representation of subject domains that deal with multiple classification levels has given rise to what has been referred to as multi-level modeling [9, 10]. In order to address the challenge of multi-level modeling, we have proposed in [11] a theory called MLT. MLT formally characterizes the nature of classification levels, and precisely defines the relations that may occur between elements of different classification levels, encompassing different notions of power type [12, 13]. Here, we apply MLT to UFO, in order to extend its applicability to domains that require multiple levels of classification. Conceptual models built with the UFO-MLT combination benefit from the ontological distinctions of UFO as well as the basic concepts and patterns for multi-level modeling of MLT.

This paper is an extension of a paper presented at the 2015 edition of the International Conference on Conceptual Modeling (ER 2015) [14]. It deals with a larger fragment of UFO, and provides a fuller treatment of modeling rules and patterns that arise from the combination of UFO and MLT. It is further structured as follows: section 2 presents a fragment of UFO and its implication to ontology-based conceptual modeling; section 3 presents the MLT multi-level modeling theory; section 4 discusses the combination of MLT and UFO to provide foundations for ontology-based multi-level modeling; section 5 identifies guidelines for multi-level conceptual modeling that arise from the foundational ontology; section 6 positions the combined foundations with respect to the existing work on multi-level conceptual modeling and finally section 7 presents concluding remarks and topics for further investigation.

2 Ontological Foundations for Conceptual Models

The Unified Foundational Ontology (UFO) is a domain independent system of categories aggregating results from disciplines such as Analytical Philosophy, Cognitive Science, Philosophical Logics and Linguistics. Over the years, UFO has been successfully employed to analyze all the classical conceptual modeling constructs including Object Types and Taxonomic Structures, Part-Whole Relations, Intrinsic and Relational Properties, Weak Entities, Attributes and Datatypes, etc. [3, 7]. Here we present a fragment of UFO that is relevant for this article. An in-depth discussion, formal characterization and discussion regarding empirical support for UFO's categories see [3].

2.1 Key Concepts

UFO begins with a distinction between *universals* and *individuals*. Universals are patterns of features that can be realized in a number of individuals. For example, "John" and "Mary" are individuals that instantiate the universals "Man" and "Woman" respectively. UFO includes a taxonomy of individuals and a taxonomy of universals.

The topmost distinction in the taxonomy of individuals is that between endurants and events. Endurants (as opposed to events) are the individuals said to be wholly present whenever they are present, i.e., they can endure in time, suffering a number of qualitatively changes while maintaining their identity (e.g., a house, a person). Since in this paper we are especially interested in a portion of UFO that accounts for structural (as opposed to dynamic) aspects of conceptual modeling, we focus solely on endurants. Endurants are further classified into *Substantials* and *Moments*. *Substantials* are existentially independent endurants (e.g. a person, a forest). A moment, in contrast, is an endurant that *inheres in*, and, therefore, is existentially dependent of, another endurant(s). Moments that are dependent of one single individual are *Intrinsic Moments* (e.g. a person's age) whereas moments that depend on a plurality of individuals are instances of *Relator* (e.g. a marriage, an employment, an enrollment).

Intrinsic moments in UFO are further classified into: (i) those that are measurable and directly related to some quality structure are termed *Qualities* (e.g. a car's weight has a measurable value in a one-dimensional structure of positive numbers). In contrast, intrinsic moments not directly related to measure structures are termed *Modes* (e.g., a person's skills, intentions, beliefs or symptoms).

These distinctions among individuals are reflected in the taxonomy of universals. Instances of *Quality Universals* have *qualities* as instances (e.g., the quality universal "Age" is instantiated by "Mick Jagger's age", "John's age"), instances of *Mode Universals* have *modes* as instances (e.g., "Disease" is instantiated by "John's diabetes", "Mary's hemophilia"), instances of *Relator Universal* have *relators* as instances (e.g., "Marriage" is instantiated by "John and Mary's Marriage"), and instances of *Substantial Universals* have substantials as instances (e.g., "Person" is instantiated by "John", "Mary", "Mick Jagger").

The ontological category of Substantial Universal is further specialized according to the ontological notions of identity and rigidity. Substantial universals that carry a uniform principle of identity for their individuals are instances of *Sortal Universal* (e.g., "Person", "Car", "Organization"). In contrast, instances of *Mixin Universal* (or *Non-Sortal Universal)* represent an abstraction of properties that are common to instances of various sortals (e.g., the *mixin* "Insurable Item" describes properties that are common to entities of different sortals such as "House", "Car", "Work of Art"). Moreover, a universal is said to be *rigid* if it classifies its instances necessarily (in the modal sense). In other words, if a universal *T* is rigid, then an instance *x* of *T* cannot cease to be an instance of *T* without ceasing to exist (e.g., "Person", "Organization"). In contrast, a universal is anti-rigid if its instances can move in and out of the extension of that universal without ceasing to exist (e.g., "Student", "Insured Organization")[1].

A *Rigid Sortal* that supplies a principle of identity to its instances is termed *Kind* (e.g., "Person", "Organization"). Instances of Kind may be specialized by other rigid sortals that inherit the principle of identity supplied by the Kind. These rigid sortals are termed *Subkinds* (e.g., "Man", "Woman"). Anti-rigid sortals are further classified into the categories *Role* or *Phase*. Instances of *Role* classify substantials through the relational properties they bear in the scope of a relational context (e.g., "Employee", "Husband", "Student"), and thus are considered externally dependent universals. In contrast, instances of *Phase* classify substantials depending on one or more of their intrinsic properties (e.g., "Child" and "Adult").

Rigid *mixins* that represent abstractions of properties that apply necessarily to instances of different kinds are called *Category* universals (e.g., "Legal Entity" abstracting properties of persons and organizations). Analogously to *anti-rigid sortals*, *anti-rigid mixins* are classified into *Role Mixin* (which are externally dependent) and *Phase Mixin*. *Role Mixins* classify substantials of different kinds through common relational properties (e.g. "Customer", abstracting relational properties applicable to persons and organizations). In contrast, *Phase Mixins* classify substantials of different kinds through common intrinsic properties (e.g. "Living Animal" and "Deceased Animal" that classify in a contingent manner instances of entities of kinds such as "Dog", "Person", "Cow", etc.). Typically, *phase mixins* appear in models forming partitions of a *category*.

In UFO, the relation between an intrinsic moment universal (a quality universal or mode universal) and a substantial universal is called *characterization* [3]. For example, to capture that every person has an age, we may define a *characterization* relation between the substantial universal "Person" and the moment universal "Age". In its turn, a relator universal is connected to substantial universals through the UFO notion of *mediation* [3]. For example, we may capture the relationship between employees and employers by defining a relator universal "Employment" and linking it to the substantial universals (roles) "Employee" and "Employer" through mediations.

Fig. 1 summarizes the discussion so far by depicting a fragment of UFO's taxonomy of universals in the left-hand side ("Endurant Universal" and its specializations) and the taxonomy of individuals in the right-hand side ("Endurant" and its specializations). This fragment of the UFO ontology is presented here as a UML class diagram for presentation-purposes only. The actual representation of the ontology is captured in [3] in a particular type of Intensional Modal Logics with Sortal Quantification.

[1] For the sake of simplicity, hare we do not consider semi-rigid types, i.e. types that classify some of its instances necessarily and some of them only contingently [15].

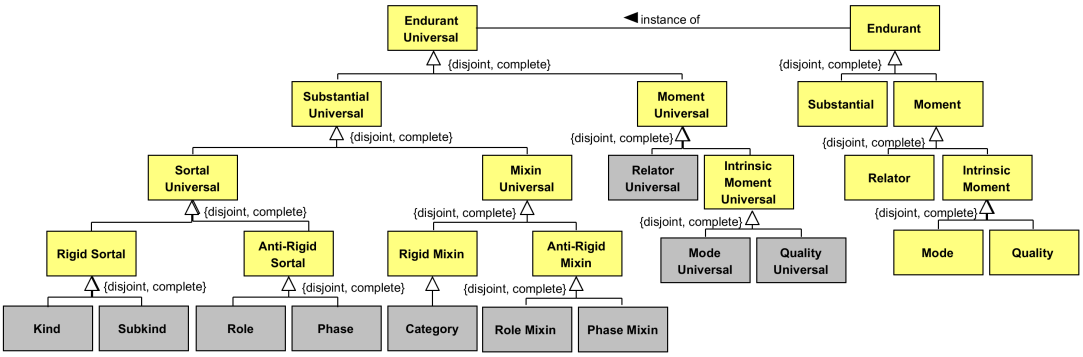

Fig. 1. UFO *endurant individuals* and *universals* taxonomies

2.2 Ontology-driven Conceptual Modeling

In order to support the construction of ontology-driven conceptual models, a UML profile for class diagrams (dubbed OntoUML) was proposed in [3]. OntoUML includes modeling primitives that reflect ontological distinctions put forth by this ontology (these are represented as stereotypes for each of the leaf ontological categories of the UFO taxonomy of universals). Over the years, it has been adopted by many research, industrial and government institutions worldwide, in areas ranging from Geology to Organ Donation, from Biodiversity Management to Logistics, from Software Engineering to Telecommunications [7]. In particular, it has been considered as a candidate for addressing the OMG SIMF (Semantic Information Model Federation) Request for Proposal, after a report of its continuous successful use by a branch of the U.S. Department of Defense [8]. Fig. 2 depicts an OntoUML diagram capturing some domain concepts previously used to exemplify the key concepts of UFO.

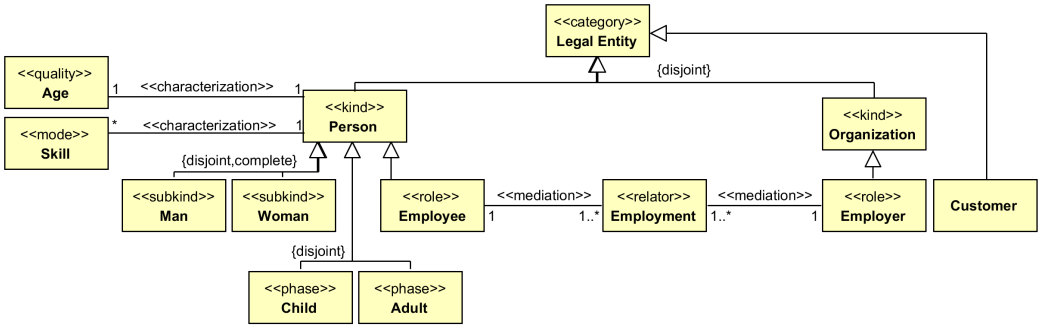

Fig. 2. An OntoUML diagram

In addition to prescribing specialized modeling primitives, OntoUML includes formal constraints that govern how these primitives can be combined. These constraints are derived from the axiomatization of the foundational ontology, which takes into account the ontological notions of *existential dependence, rigidity and principle of identity*, among others. We present here a subset of these formal constraints, namely those that govern how specialization between domain types may be applied to ensure that the resulting conceptual models are sound.

Considering that moments and substantials are disjoint, since no entity can be at the same time existentially-dependent and existentially-independent, instances of *Moment Universal* may not specialize instances of *Substantial Universal* and vice-versa (*rule U1*). The same applies to qualities, modes and relators; these are disjoint ontological categories, and thus instances of *Quality Universal*, *Mode Universal* and *Relator Universal* cannot specialize each other (*rule U2*).

In the case of substantial universals, a further rule arises from the distinction into sortals and mixins (i.e., non-sortals): instances of *Mixin Universal* cannot specialize instances of *Sortal Universal* (*rule U3*). This is because if a mixin universal were to specialize a sortal universal, it would inherit the principle of identity carried by that sortal universal. However, by definition, mixin universals do not carry a principle of identity.

A further constraint arises from the distinction into rigid and anti-rigid universals: rigid universals (instances of *Rigid Sortal* and *Rigid Mixin*) may not specialize instances of anti-rigid universals (*Anti-*

Rigid Sortal and *Anti-Rigid Mixin*) (*rule U4*). If a rigid subtype were to specialize an anti-rigid supertype, its instances would instantiate the subtype necessarily (by definition) and would also instantiate the supertype necessarily (by virtue of specialization). However, as we considering an anti-rigid supertype – which applies contingently (i.e., non-necessarily) to its instances – this is not possible by definition. Note that the converse is admissible, and thus an anti-rigid type may specialize a rigid one as no contradiction arises.

Two rules emerge from the source of principle of identity in sortal specialization. First, an instance of *Kind* cannot specialize another sortal universal (*rule U5*). A kind K supplies a principle of identity to its instances, and, if it were to specialize another sortal universal S, it would inherit a principle of identity carried by S, i.e., supplied by S or by another kind that S specializes. As a consequence, K, in this case, would carry more than one principle of identity for its instances. As discussed in depth in [3], these instances would become ontologically indeterminate individuals. Second, instances of *Subkind*, *Phase* and *Role* are sortals that carry – but do no supply – a principle of identify for their instances. Hence, they must specialize a unique instance of *Kind* (*rule U6*), which will supply the principle of identity for their instances.

3 MLT: A Theory for Multi-Level Modeling

Conceptual domain models constructed in OntoUML are able to express ontological properties of the types that apply to individuals in the subject domain. However, currently, no support is provided to represent domain-specific types of types, since the second-order types of OntoUML are predefined in the language profile (as stereotypes). This motivates our investigation into the combination of UFO with a theory capable of accounting for multiple levels of classification.

We employ a theory for multi-level modeling called MLT, which we introduced in [11]. Similarly to UFO, MLT distinguishes between types (universals) and individuals. However, differently from UFO, MLT also considers types that have other types as instances. In order to accommodate these varieties of types, the notion of *type order* is used in MLT. Types having individuals as instances are called *first-order types,* types whose instances are first-order types are called *second-order types,* and so on.

In order to link types to the entities that fall under such types, MLT defines a primitive *instance of* relation. This relation is represented by a ternary predicate *iof(e,t,w)* that holds if an entity *e* is instance of an entity *t* (denoting a type) in a world *w*. Indexing the instantiation relation to possible worlds allows MLT to support *dynamic classification*, admitting thus types that apply contingently to their instances (e.g., "John" is an instance of "Student" in *w* but not in *w'*, when he has graduated.)

The axiomatic theory is built up defining the conditions for entities to be considered individuals, using the logic constant "Individual". Thus, an entity is an instance of "Individual" iff it does not have any possible instance. The constant "First-Order Type" (or shortly "1stOT") characterizes the type that applies to all entities whose instances are instances of "Individual". Analogously, each entity whose possible extension contains exclusively instances of "1stOT" is an instance of "Second-Order Type" (or shortly "2ndOT"). It follows from this definition that "Individual" is instance of "1stOT" which, in turn, is instance of "2ndOT". We call "Individual", "1stOT" and "2ndOT" the basic types of MLT. According to MLT, every possible entity must be instance of exactly one of its *basic types* (except the topmost type)[2]. For our purposes in this paper, *first-* and *second-order types* are enough. However, this scheme can be extended to consider as many orders as necessary.

Fig. 3 illustrates the elements that form the basis for our multi-level modeling theory, using a notation that is largely inspired in UML. We use the UML class notation to represent the *basic types of* MLT. We use associations as usual to represent relations between instances of the related types (predicates that may be applied to instances of the related types). We use dashed arrows to represent relations that hold between the types, with labels to denote the names of the predicates that apply. This notation is used in all further diagrams in this paper. It is important to highlight here that our focus is

[2]One should notice that, in MLT, anything that is not an individual is a type of a given order. According to this definition of individual, we have that a type is anything that can possibly have an instance. This definition of type conforms to the so-called *Principle of Instantiation* required for universals in the philosophical literature [16].

not on the syntax of a multi-level modeling language and that we use these diagrams to illustrate the concepts intuitively. A complete formalization of MLT can be found in [11].

Fig. 3. Foundations of MLT: *basic types* and the *instance of* relation.

Some structural relations to support conceptual modeling are defined in MLT, starting with the ordinary specialization between types. A type *t specializes* another type *t'* iff in all possible worlds all instances of *t* are also instances of *t'*. According to this definition every type *specializes* itself. Since this may be undesired in some contexts, we define the *proper specialization* relation as follows: *t proper specializes t'* iff *t specializes t'* and *t* is different from *t'*. Note that the definitions presented thus far guarantee that both *specializations* and *proper specializations* may only hold between types of the same order (these relations are depicted in the upper part of Fig. 4).

Every type that is not a basic type (e.g., a domain type) is an instance of one of the basic higher-order types (e.g., "1stOT", "2ndOT"), and, at the same time proper specializes the basic type at the immediately lower level (respectively, "Individual" and "1stOT"). Fig. 4 illustrates this pattern. Since "Person" applies to individuals, it is instance of "1stOT" and proper specializes "Individual". The instances of "Person Age Phase" are specializations of "Person" (e.g. "Child" and "Adult"). Thus, "Person Age Phase" is instance of "2ndOT" and proper specializes "1stOT". This pattern will be applied to UFO concepts in Section 4 and will drive rules for the introduction of second-order types in domain models in Section 5.

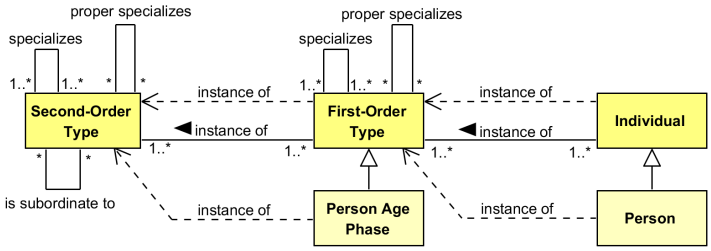

Fig. 4. Illustrating an important basic pattern of MLT and its intra-level structural relations.

In addition to the instantiation and specialization relations, MLT also defines a *subordination* relation. *Subordination* between two higher-order types implies *specializations* between their instances i.e., *t is subordinate to t'* iff every *instance of t proper specializes* an *instance of t'*. Since *subordination* implies *proper specializations* between the instances of the involved types at one order lower, subordination can only hold between higher-order types of equal order. We will use subordination to explain the relation between universals in UFO's taxonomy of universals (e.g., since every "Subkind" must specialize a "Kind", "Subkind" *is subordinate to* "Kind".)

So far, we have only considered intra-level relations, i.e., those that occur between entities of the same order. In addition to that, MLT defines *cross-level structural relations* between types of adjacent orders. These relations support an analysis of the notions of power type in the literature, leading to their incorporation in the theory.

Based on the notion of *power type* proposed by Cardelli [12], which is founded on the notion of powerset, MLT defines a *power type* relation between a higher-order type and a base type at an order lower: a type *t is power type of* a *base type t'* iff all instances of *t specialize t'* and all possible *specializations of t' are instances of t*. Thus, a type *t is the powertype of t'* when the instances of *t* are applicable to instances of *t'* but *t* does not define *classification criteria*. Thus, all specializations of *t'*, including *t'* itself are instances of *t*. For example, consider a type called "Person Powertype" such that all possible specializations of "Person" are instances of it and, conversely, all its instances specialize "Person". In this case, "Person Powertype" is the *power type* of "Person". Since "Person" is instance of "1stOT", "Person Powertype" is instance of "2ndOT" and specializes "1stOT" (see Fig. 5). (It follows from the definition of *power type* that "1stOT" is *power type of* "Individual". Analogously, "2ndOT" is *power type of* "1stOT".).

Another definition of *power type* that has had great influence in software engineering was proposed by Odell [13]. Inspired on Odell's definition [13], MLT defines the *categorization* relation between

types of adjacent levels: a type *t categorizes* a type *t'* iff all *instances of t* are *proper specializations* of *t'*. The *categorization* relation occurs between a higher order type t and a base type t' when all instances of t specialize t' according to specific *classification criteria*. Thus, the instances of t specialize t' but t' is not an instance of t and there may be other types that specializes t' according to other *classification criteria* and, thus, are not instances of t. For instance in Fig. **5**, "Person Role" (with instances "Manager" and "Researcher") *categorizes* "Person", but is not a *power type of* "Person", since there are specializations of "Person" that are not instances of "Person Role" ("Child" and "Adult" in the example).

We also define some variations of *categorization*, which are useful to capture further constraints in a multi-level model. We consider that a type *t completelyCategorizes t'* iff *t categorizes t'* and every instance of *t'* is *instance of,* at least, an *instance of t.* Moreover, iff *t categorizes t'* and every instance of *t'* is *instance of,* at most, one *instance of t* it is said that *t disjointlyCategorizes t'*. Finally, a common use for the notion of *power type* in literature considers a *second-order type* that, simultaneously, *completely* and *disjointly categorizes* a *first-order type*. To capture this notion we defined the *partitions* relation. Thus, *t partitions t'* iff *t categorizes t' and* each *instance of t'* is *instance of* exactly one *instance of t.* For example of the partitioning relation, consider the second-order type called "Person Age Phase" with instances "Child" and "Adult" (Fig. 5). (This kind of constraint is usually represented in UML through a generalization set, see [37] for a detailed comparison).

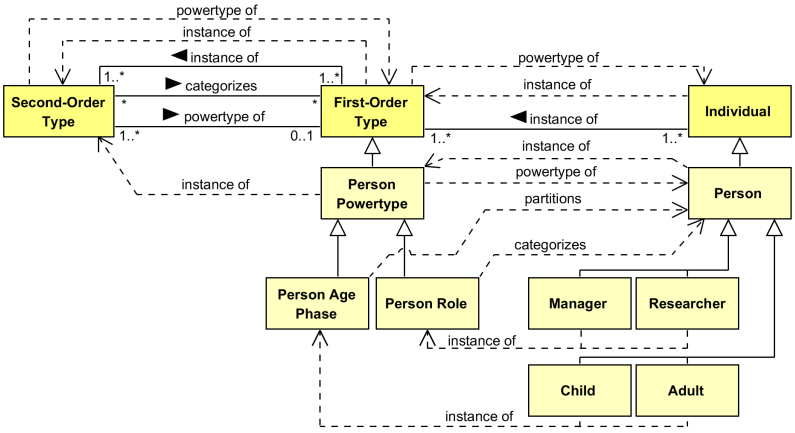

Fig. 5**.** Illustrating the use of MLT for multi-level conceptual modeling.

Table 1 summarizes the semantics of each MLT cross-level relation. All these relations are irreflexive, antisymmetric and anti-transitive.

Table 1. Cross-level structural relations semantics

Name	Meaning
Power type of *isPowertypeOf(t, t')*	The instances of *t* apply to instances of *t'* but no *classification criteria* are defined. Thus, all specializations of *t',* including *t'* itself are instances of *t*.
Categorization *categorizes(t, t')*	The instances of *t* apply to instances of *t'* according specific *classification criteria*. Thus, all proper specializations of *t'* that follow the specified *classification criteria* are instances of *t*.
Complete Categorization *completelyCategorizes(t, t')*	A variation of *categorization* in which the *classification criteria* defined by *t* guarantee that each instance of *t'* is instance of at least one instance of *t*.
Disjoint Categorization *disjointlyCategorizes(t, t')*	A variation of *categorization* in which the *classification criteria* defined by *t* guarantee that each instance of *t'* is instance of at most one instance of *t*.
Partitioning *partitions(t, t')*	A variation of *categorization* in which the *classification criteria* defined by *t* guarantee that each instance of *t'* is instance of exactly one instance of *t*.

4 Combining MLT and UFO

In order to combine MLT and UFO, we establish a hierarchy of conceptual models, with MLT forming the topmost layer. The basic pattern of MLT is applied throughout the hierarchy, first to establish the relation between MLT and UFO, and later to establish the relation between a conceptual domain

model and UFO-MLT. More specifically, the concepts of UFO *instantiate* and *specialize* elements of MLT, thereby respecting MLT's axioms and leveraging the use of structural relations and patterns of MLT in UFO. In turn, the concepts of the conceptual domain models *instantiate* and *specialize* concepts of MLT and UFO, respecting all rules and patterns of both MLT and UFO.

The concepts in UFO's taxonomy of individuals are instances of "1stOT" *specializing* "Individual". The concepts in the taxonomy of universals are instances of "2ndOT" *specializing* "1stOT". For each entity in the taxonomy of individuals (e.g., "Endurant", "Substantial", "Moment"), there is a corresponding entity in the taxonomy of universals ("Endurant Universal", "Substantial Universal", "Moment Universal"). Instances of the entity in the taxonomy of universals specialize the corresponding entity in the taxonomy of individuals. Thus, "Endurant Universal" *categorizes* "Endurant", "Substantial Universal" *categorizes* "Substantial", and so on.

In addition to these general categorization relations, we can also use more specific MLT relations to further explain how the two taxonomies relate according to ground notions in UFO (such as identity). For example, since each instance of "Substantial" is an instance of exactly one instance of "Kind" (the kind that supplies the principle of identity), following MLT, "Kind" partitions "Substantial". In addition, since they carry (but do not supply) a principle of identity, instances of "Subkind", "Phase" and "Role" must specialize an instance of "Kind" that supplies such principle (*rule U6*). Thus, in MLT terms, "Subkind", "Phase" and "Role" are subordinate to "Kind". Fig. 6 illustrates the resulting two-layer hierarchy revealing these relations.

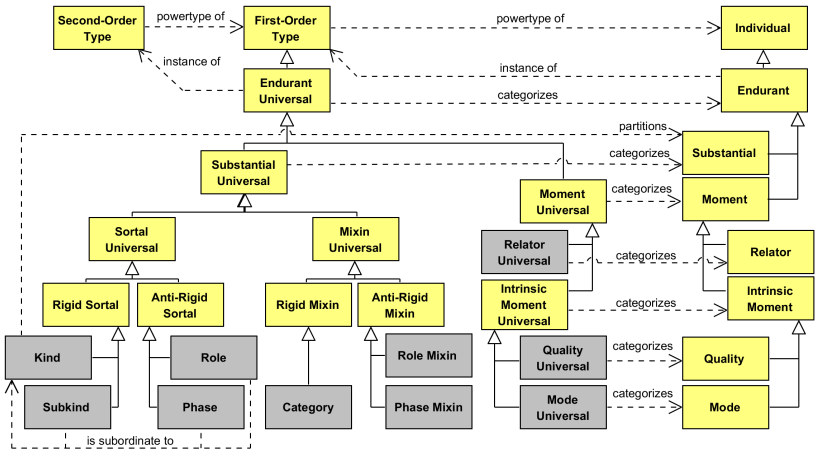

Fig. 6. Applying MLT to the UFO fragment on endurants

In order to benefit from the ontological distinctions of UFO as well as the basic concepts and patterns for multi-level modeling of MLT, conceptual models built with the UFO-MLT combination must adhere to the rules of both theories. Thus, every domain first-order type must: (i) instantiate one of the leaf ontological categories of UFO's taxonomy of universals (and, consequently, instantiate MLT's "1stOT"); and (ii) simultaneously, specialize one of the leaf ontological categories of UFO's taxonomy of individuals (and thus, specialize "Individual"). For example, consider "Legal Entity" as instance of "Category". Since "Category" specializes "1stOT" we conclude that "Legal Entity" is also a first-order type. Considering that "Category" categorizes "Substantial", it follows that "Legal Entity" specializes "Substantial" (and, indirectly, specializes "Individual"). Analogously, "Person" and "Organization" are instances of "Kind" and specialize "Legal Entity" (indirectly specializing "Substantial" and "Individual").

In our approach, we introduce domain *second-order* types as specializations of one of the leaf categories of UFO's taxonomy of universals. Following [11], in order to clarify which first-order type is ultimately instantiated by instances of a second-order type, every domain second-order type has an MLT cross-level relation (i.e., *categorizes*, *disjointly categorizes*, *completely categorizes* or *partitions*) with a first-order type, which is termed its *base type.* Further, we consider that a second-order type also defines the *classification criteria* that its instances must apply to specialize the *base type*. As a result, we can use the UFO-MLT combination to provide rules and patterns for introducing *second-order types* in ontology-based domain models. This is the subject of section 5.

5 Second-Order Types in Ontology-based Domain Models

This section discusses the introduction of second-order types in ontology-based domain conceptual models according to the UFO leaf categories being specialized. For each specialization case, we identify the admissible ontological categories for a (first-order) base type as well as the MLT cross-level relations that may be established between the second-order type and the base type. This is possible by combining the MLT basic pattern, which determines that each instance of a second-order type specializes a (first-order) base type (as discussed in section 3), with UFO rules that determine constraints for specialization relations according to the types' ontological categories (as discussed in section 2).

As a general rule, we have that substantial universals cannot specialize moment universals and vice-versa (*rule U1*), and thus a specialization of "Substantial Universal" cannot have an instance of "Moment Universal" as a base type. Conversely, a specialization of "Moment Universal" cannot have an instance of "Substantial Universal" as a base type. Specific rules that apply to second-order types specializing each leaf category of the UFO taxonomy of universals are considered in the sequence. First, in section 5.1, we consider second-order types specializing "Sortal Universal" (viz., the leaf categories "Kind", "Subkind", "Phase" and "Role"). Second, in section 5.2, we consider second-order types specializing "Mixin Universal" (viz., the leaf categories "Category", "Phase Mixin", "Role Mixin"). Finally, in section 5.3, we consider second-order types specializing "Moment Universal" (viz., "Quality Universal", "Mode Universal" and "Relator Universal") and discuss the relation between taxonomies of moments and taxonomies of substantials.

5.1 Second-Order Types Specializing "Sortal Universal"

Specializations of Kind. According to UFO, kinds are sortal universals that supply an identity criteria to their instances, and hence, they cannot specialize another sortal universal (*rule U5*). This rules out sortal universals as base types for second-order types *specializing* "Kind". Further, considering that kinds are *rigid universals*, they cannot specialize *anti-rigid universals* (*rule U4*), which also rules out anti-rigid universals as base types for specializations of "Kind". Therefore, by excluding sortal universals and anti-rigid universals as admissible base types, a second-order type *specializing* "Kind" must have a *rigid mixin sortal* as *base type* (i.e., an instance of "Category").

Moreover, considering that each individual must be instance of exactly one kind, we can more specifically state that a specialization of "Kind" must *partition* an instance of "Category", using as *classification criteria* the principle of identity supplied by the kind. For example, considering "Legal Entity" as a "Category" that generalizes properties of different kinds of legal entities, we may define a *second-order type* "Legal Entity Kind" that *specializes* "Kind" and *partitions* "Legal Entity" having as instances the kinds "Person" and "Organization". Fig. 7 illustrates this scenario.

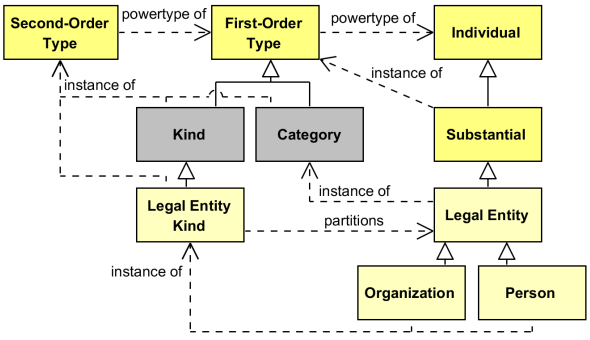

Fig. 7. A domain second-order type specializing "Kind" and partitioning an instance of "Category"

Specializations of Subkind. Since subkinds are *rigid universals*, they cannot specialize instances of *anti-rigid universals* (*rule U4*). Thus, a *second-order* type specializing "Subkind" must have, as *base type*, an instance of "Rigid Sortal" (i.e., an instance of "Kind" or "Subkind") or an instance of "Rigid Mixin" (i.e., an instance of "Category").

Subkinds are common in taxonomies in which the more specific types form a partition of a more general type distinguishing instances according to immutable intrinsic properties (e.g., "Person" spe-

cialized into “Man” and “Woman” according to gender). In these cases, a second-order type that *specializes* “Subkind” *partitions* an instance of “Kind”, “Subkind” or “Category” using some immutable intrinsic properties exemplified by their instances as classification criteria (see “Person Gender Subkind” with instances “Man” and “Woman” in Fig. 8).

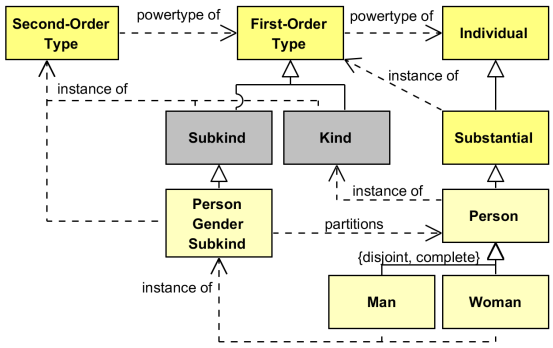

Fig. 8. A domain second-order type specializing “Subkind” and partitioning an instance of “Kind”.

Another example of *second-order type* that can be represented as a specialization of “Subkind” is “Horse Breed”, *partitioning* “Horse” having instances such as “Mustang Horse” and “Arabian Horse”. Note that MLT does not force the modeler to enumerate the instances of second-order types (such as “Horse Breed”), while still capturing the fact that its instances form a partition of the base type (“Horse”) (see Fig. 9).

Differently from the cases in which the base type is a kind or a subkind, when the base type is a category, in addition to specializing the base type, the instance of the higher-order type must also specialize some instance of “Kind”. This is a consequence of *rule U6*, which was captured through the *subordination* of “Subkind” to “Kind” in the UFO-MLT combination. In these cases, the principle of identity carried by the instances of the second-order type is not yet settled by the base type, and thus the principle of identity must be determined for each instance of the second-order type. In the example of Fig. 9, the *second-order type* “Female Animal Subkind” *specializes* “Subkind” and *partitions* “Female Animal” having instances such as “Cow” and “Mare” (see Fig. 9). “Cow” *specializes* “Ox”, inheriting its principle of identity, while “Mare” *specializes* “Horse”, inheriting thus a different principle of identity.

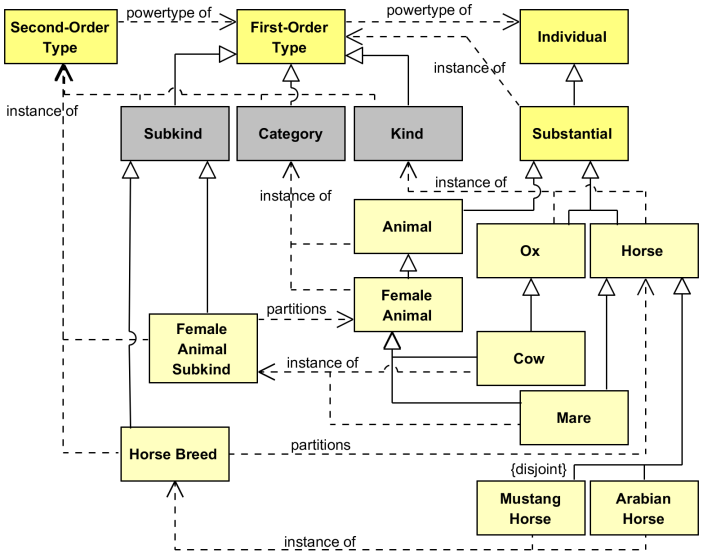

Fig. 9. Domain second-order types specializing “Subkind” and partitioning instances of “Category” and “Kind”.

Specializations of Phase. According to UFO, instances of “Phase” are anti-rigid sortal universals. As such, they may specialize any substantial universal. A *second-order type* that *specializes* “Phase” may thus have as *base type* any instance of “Substantial Universal” regardless of rigidity and sortality (i.e., an instance of “Kind”, “Subkind”, “Phase”, “Role”, “Category”, “Phase Mixin” or “Role Mixin”).

As discussed in [3], instances of "Phase" classify individuals of a specific kind using some mutable intrinsic property as classification criteria, forming partitions of a more general type. We can capture this notion with a *second-order type* that specializes "Phase" and *partitions* an instance of "Substantial Universal". For example, in Fig. 10, "Person Age Phase" *partitions* "Person" into "Child", "Adult" and "Elder" according to the intrinsic property age.

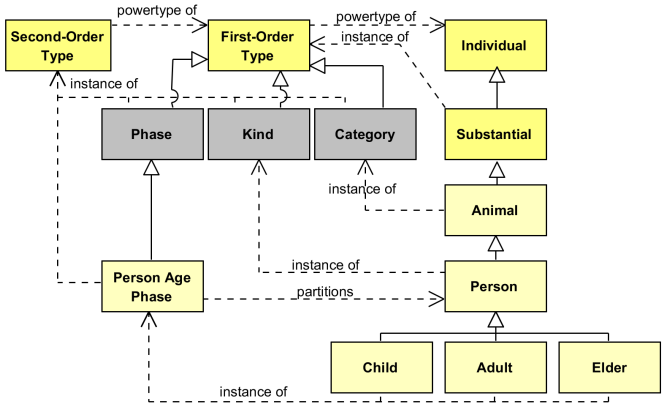

Fig. 10. A domain second-order type specializing "Phase" and partitioning an instance of "Kind".

To capture domains in which some mutable intrinsic properties are applicable to individuals of different kinds, one may define *specializations* of "Phase" having as base types instances of "Mixin Universal" (i.e. "Category", "Phase Mixin" and "Role Mixin"). In these cases, the principle of identity carried by the instances of the second-order type is determined for each instance, by establishing the kind that it specializes. In the example in Fig. 11, the second-order type "Young Animal Phase" *partitions* "Young Animal" having "Child" and "Calf" as instances; "Child" *specializes* the kind "Person", inheriting its principle of identity, while "Calf" *specializes* the kind "Ox", inheriting thus a different principle of identity.

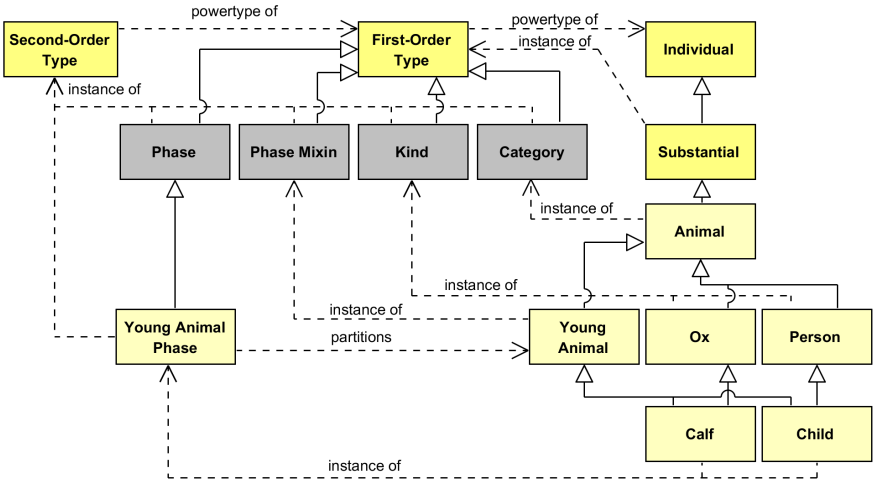

Fig. 11. A domain second-order type specializing "Phase" and partitioning an instance of "Phase Mixin".

Specializations of Role. Similarly to instances of "Phase", instances of "Role" are anti-rigid sortal universals and may specialize any substantial universal. Thus, a *second-order type* that *specialize* "Role" may have as *base type* any instance of "Substantial Universal" (i.e., an instance of "Kind", "Subkind", "Phase", "Role", "Category", "Phase Mixin" or "Role Mixin").

As discussed in [3], instances of "Role" classify individuals through the relational properties they bear in the scope of a relational context. A *second-order type* specializing "Role" may thus *categorize* an instance of "Substantial Universal" using as classification criteria some relational properties exemplified by the instances of the base type. For example, we can define a *second-order type* "Person Role" that *categorizes* "Person" according to roles that persons may play during their lives. Types such as "Employee", "Driver" and "Wife" would be examples of instances of "Person Role". More specific specializations of "Person Role" include: (i) "Woman Role" whose instances specialize

"Woman" (an instance of "Subkind") and include those roles that are played exclusively by women, such as "Wife" or "Biological Mother"; (ii) "Adult Role" whose instances specialize "Adult" (an instance of "Phase") and include those roles that are played exclusively by adults, such as "Driver" and "Congressman"; and, (iii) "Employee Type" whose instances specialize "Employee" (an instance of "Role") into "Temporary Employee" and "Permanent Employee", considering the type of work contract they have. These examples of second-order types specializing "Role" are illustrated in Fig. 12.

To capture domains in which roles can be played by individuals of different kinds, one may define second-order types that *specialize* "Role" having as base types instances of "Mixin Universal" (i.e., instances of "Role Mixin", "Phase Mixin" or "Category"). In these cases, each instance of the second-order type must specialize a sortal universal to settle the principle of identity. For example, considering that "Customer" is a "Role Mixin" that generalizes a role that can be played by any "Legal Entity", we may define a *second-order type* "Customer Type" that *specializes* "Role" and *partitions* "Customer" having as instances "Personal Customer" and "Corporate Customer". "Personal Customer" *specializes* "Person", inheriting its principle of identity, while "Corporate Customer" *specializes* "Organization", inheriting thus a different principle of identity (see Fig. 13).

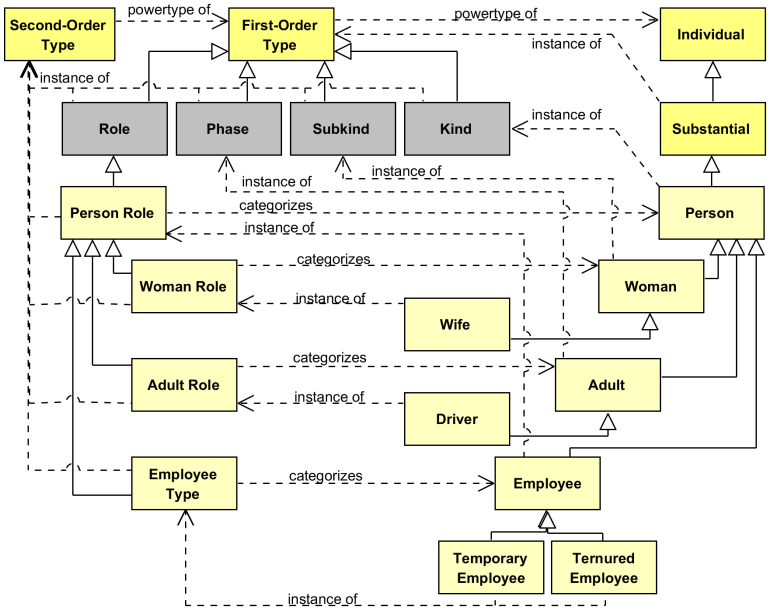

Fig. 12. Domain second-order types specializing "Role" and *categorizing* instances of "Sortal Universal"

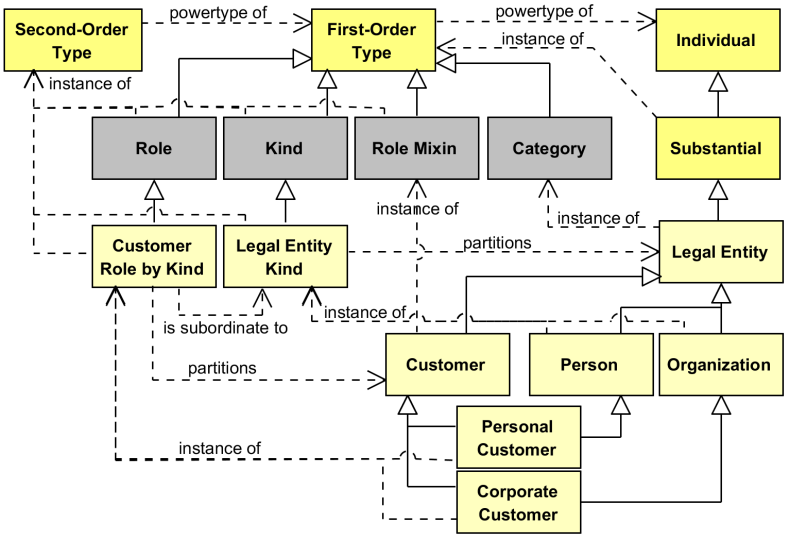

Fig. 13. Domain second-order types specializing "Kind" and "Role" and *partitioning* instances of "Mixin Universal"

5.2 Second-Order Types Specializing "Mixin Universal"

Similarly to "Sortal Universal", "Mixin Universal" is a specialization of "Substantial Universal", and thus, its instances cannot specialize instances of "Moment Universal" (*rule U1*). Moreover, considering that mixin universals cannot specialize sortal universals (*rule U3*), we have that second-order types specializing "Mixin Universal" cannot have an instance of "Sortal Universal" as base type. Therefore, a specialization of "Mixin Universal" must *categorize* an instance of "Mixin Universal", ultimately an instance of "Category", "Phase Mixin" or "Role Mixin". Specific rules that apply to second-order types specializing each leaf category of mixin universal are considered in the sequence.

Specializations of Category. Since instances of "Category" are *rigid universals*, they may not specialize *anti-rigid universals* (*rule U4*). Therefore, a *second-order type* that *specialize* "Category" must have, as *base type,* a rigid mixin universal, i.e. an instance of "Category". This structure gives rise to hierarchies of categories in which the more specific types usually form a partition of a more general type distinguishing instances according to some immutable intrinsic properties. In these cases, the *second-order type specializing* "Category" *partitions* an instance of "Category". For example, considering "Animal" is a category that generalizes intrinsic immutable properties of different kinds of animals, we may define a *second-order type* "Animal Category by Presence of a Backbone" that *partitions* "Animal" having as instances "Vertebrate Animal" and "Invertebrate Animal" (see Fig. 14).

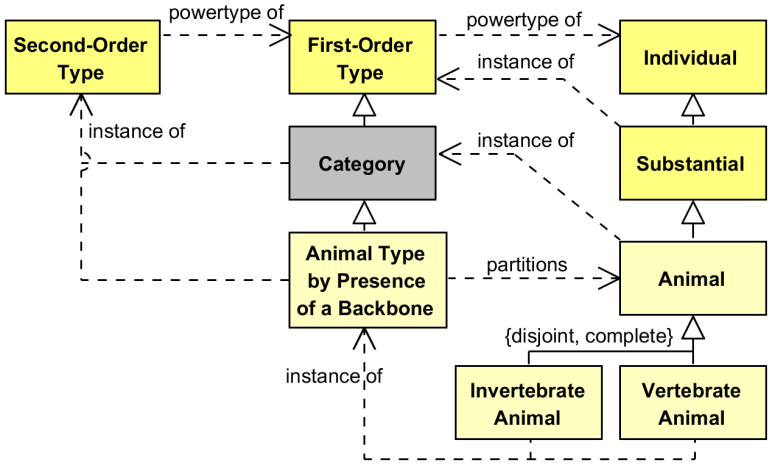

Fig. 14. A domain second-order type specializing "Category" and *partitioning* an instance of "Category"

Other examples of second-order types specializing "Category" can be found in the biological taxonomy for living beings. Such taxonomy classifies living beings according to biological taxa in seven or more ranks, e.g., kingdom, phylum, class, order, genus, species, and breed. As discussed in [11], the seven biological ranks are *second-order types* that *partition* "Living Being" obeying a subordination chain such that "Phylum" *is subordinate to* "Kingdom", "Class" *is subordinate to* "Phylum", and so on. Assuming, that the identity criteria for living beings are provided by their species, we have that "Species" specializes "Kind" and *partitions* "Living Being". In this case, since specializations of "Kind" may only be subordinate to specializations of "Category", the five biological ranks to which "Species" *is subordinate*, namely "Kingdom", "Phylum", "Class", "Order" and "Genus", are *specializations* of "Category". "Breed", in its turn, is *subordinate to* "Species" and *specializes* of "Subkind". This scenario is illustrated in Fig. 15.

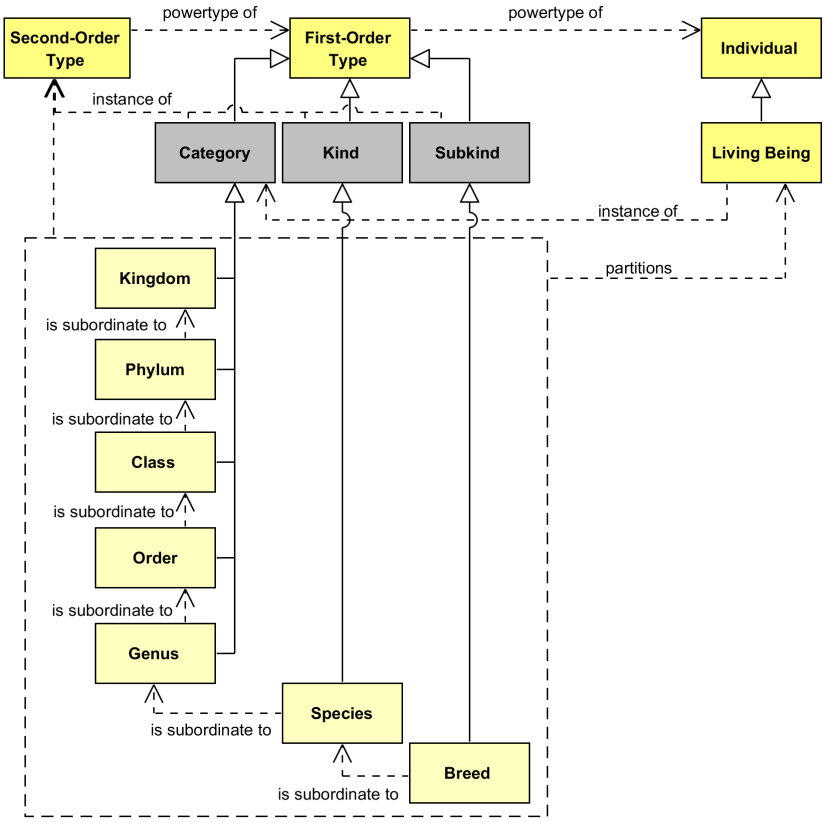

Fig. 15. Using MLT-UFO combination to describe concepts involved in biological taxonomy

Specializations of Phase Mixin. Instances of "Phase Mixin" are anti-rigid mixin universals. As such, they may specialize any mixin universal. Thus, a *second-order type* that specializes "Phase Mixin" may have as *base type* any instance of "Mixin Universal (i.e. an instance of "Category", "Phase Mixin" or "Role Mixin"). Considering that *phase mixins* classify individuals of different kinds according to some mutable intrinsic property, the instances of "Phase Mixin" usually form partitions of a more general type. We can capture this notion with a *second-order type* that specializes "Phase Mixin" and *partitions* an instance of "Mixin Universal". For example, in Fig. 16, "Animal Phase" is a second-order type that *specializes* "Phase Mixin" and *partitions* "Animal" (an instance of "Category") having as instances "Living Animal" and "Dead Animal".

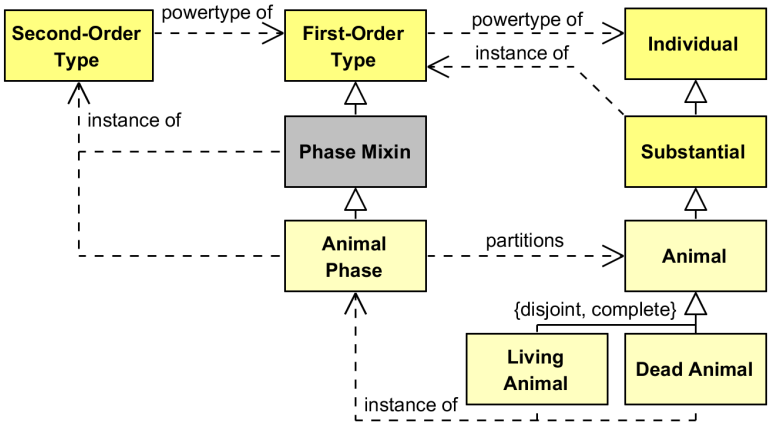

Fig. 16. Second-order types specializing "Phase Mixin"

Specializations of Role Mixin. Similarly to instances of "Phase Mixin", instances of "Role Mixin" are anti-rigid mixin universals and may only specialize other mixin universals. Thus, a *second-order type* that specializes "Role Mixin" may have as *base type* any instance of "Mixin Universal (i.e. an instance of "Category", "Phase Mixin" or "Role Mixin").

In contrast to specializations of "Phase Mixin", second-order types specializing "Role Mixin" use as classification criteria some relational properties individuals bear in the scope of a relational context, allowing us to capture scenarios in which a role can be played by individuals of different kinds. For example, considering "Legal Entity" as a "Category" that generalizes properties of different kinds of legal entities, we may define a *second-order type* "Legal Entity Role" that *specializes* "Role Mixin" and *categorizes* "Legal Entity" having instances such as "Customer" and "Supplier". The *second-order type* "Customer Role", in its turn, specializes "Role Mixin" and *categorizes* "Customer" having as instances types that defines roles that are played by customers such as "Account Holder" and "Insurance Beneficiary". This scenario is illustrated in Fig. 17

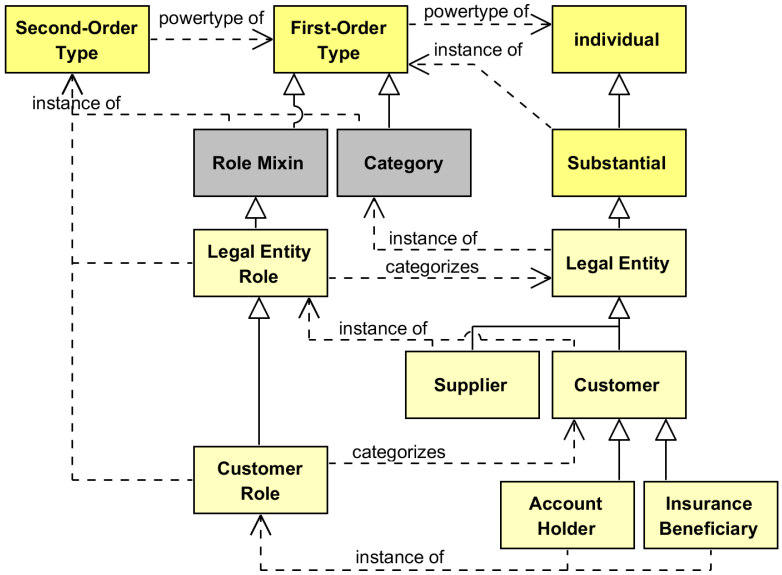

Fig. 17. Domain second-order types specializing "Role Mixin".

5.3 Second-Order Types Specializing "Moment Universal"

Thus far we have focused on second-order types specializing "Substantial Universal". As evidenced in the previous sections, these types define the criteria used by their instances to specialize a *base type* according to certain (intrinsic or relational) properties of substantials. These properties are formally accounted for in UFO using the notions of moment (particularized properties that are ultimately dependent on substantials) and moment universal (universals whose instances are moments). As a consequence, in a conceptual domain model founded on UFO, moment universals are treated as types along with substantial universals. In this section, we address the second-order types specializing each leaf category of UFO's taxonomy of moment universals, discussing rules for defining the admissible ontological category for their base types. We further explore the interrelations between moment universals and substantial universals in a domain conceptual model.

Differently from the taxonomy of substantial universals, in its original version, UFO does not elaborate on the rigidity and sortality of moment universals[3]. The UFO taxonomy of moment universals reflects the taxonomy of moment individuals, having thus three leaf concepts, namely "Quality Universal", "Mode Universal" and "Relator Universal". Considering that "Quality", "Mode" and "Relator" are disjoint types *categorized* respectively by "Quality Universal", "Mode Universal" and "Relator Universal", it is not possible for an instance of one leaf concept of the moment universal taxonomy to specialize an instance of another leaf concept (*rule U2*). Adding to this, the fact that instances of "Moment Universal" cannot specialize instances of "Substantial Universal" (*rule U1*), we conclude that specializations of each leaf concept of the moment universal taxonomy may only have instances of the same leaf concept as *base type*. Thus, specializations of "Quality Universal" must *categorize*

[3]There is a recent proposal for extending UFO in which the distinctions between sortals and mixins as well as between rigid and anti-rigid types are applied to moment universals [7, 18]. This proposal, however, has neither been formally characterized nor incorporated into results derived from UFO (e.g., OntoUML). For this reason, this proposal is not considered here.

instances of “Quality Universal”, specializations of “Mode Universal” must *categorize* instances of “Mode Universal” and specializations of “Relator Universal” must *categorize* instances of “Relator Universal”. We consider each of these cases in the sequence.

Specializations of Quality Universal. Following the theory of Conceptual Spaces [17], UFO assumes that for several perceivable or conceivable quality universals there is an associated *quality dimension* in human cognition. For example, “Age” and “Height” are associated with one-dimensional structures with a zero point isomorphic to the half-line of nonnegative numbers while “Color” can be associated to a structure (a quality domain) formed by three dimensions, named Hue, Saturation and Brightness. In Fig. 18, following [3], we use the UML construct of a structured datatype to model the color domain according to the Hue, Saturation and Brightness (HSB) scheme. In this representation, the datatype attributes “hue”, “saturation” and “brightness” are placeholders for the coordinates of each of the quality dimensions forming the color domain [3]. The separate identification of the “Color” quality universal allows us to use different quality structures if necessary, each of which with a separate corresponding datatype. Further, we can account for the change in a quality of an entity by admitting that the value of the structured datatype may change. For example, we can account for the change of the color of a particular apple from green to red, and from red to brown by admitting changes on its values in the structured data type.

Second-order types specializing “Quality Universal” can be defined to *categorize* an instance of “Quality Universal” considering possible regions of the associated quality structure. For example, we can define a second-order type “Color Type” *specializing* “Quality Universal” and *categorizing* “Color” according to selected regions of a color domain having instances such as “Blue-Toned Color” and “Green-Toned Color” (see Fig. 18). Since each instance of “Color Type” determines a region of the color domain, its instances (i.e., instances of “Color”) always have values for quality dimensions within the specified region. Determining whether the categorization is disjoint or not would allow us to represent whether there is overlap in the regions specified by the various instances of “Color Type”.

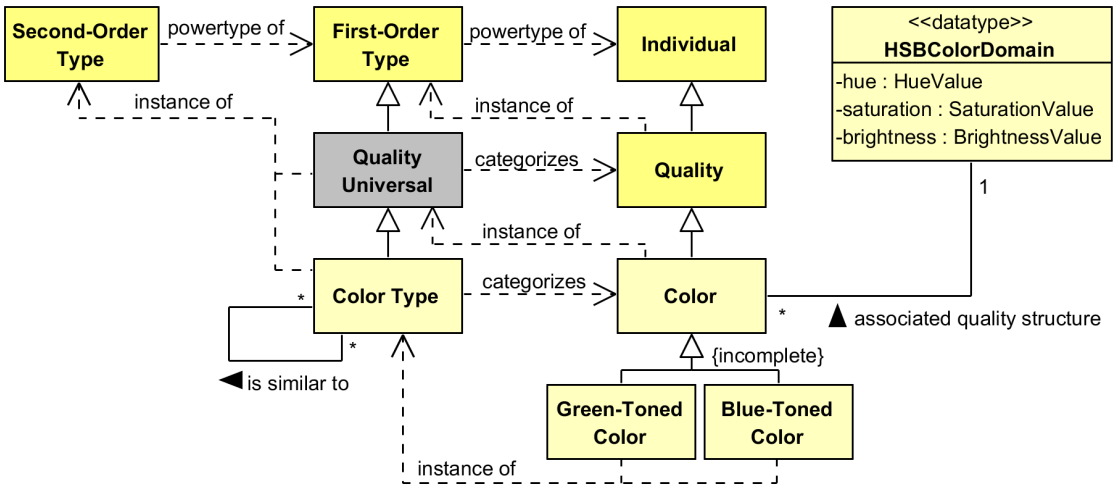

Fig. 18. Second-order types specializing “Quality Universal”.

The possibility of defining second-order types categorizing instances of Quality Universal allows us to define important relations between these instances, which appear very frequently in the characterization of quality universals. For instance, one might want to represent that a color type is similar to another one (a binary relation between instances of “Color Type”). For example, “Red-Toned Color” is similar to “Orange-Toned Color”, while “Red-Toned Color” is not similar to “Blue-Toned Color”.

As discussed in section 2, the relation between an intrinsic moment universal (a quality universal or mode universal) and a substantial universal is made explicit using the UFO notion of *characterization* [3]. For example, consider an instance of “Category” called “Physical Object” which in a particular conceptualization encompasses all entities which are tangible and visible, such as a car, a fruit, and so on. To capture that a physical object has a color, we may relate the quality universal “Color” to “Physical Object” through characterization. This means that instances of “Color” are moments inher-

ing in instances of “Physical Object”. The same criteria used to specialize a quality universal can be used to specialize the characterized substantial universal, establishing a correspondence between the taxonomy of qualities and the taxonomy of substantials. For example, we could define substantial universals such as “Blue Object” and “Green Object” specializing “Physical Object” reflecting the specialization of “Color” into “Green-Toned Color” and “Blue-Toned Color”.

In this setting, in addition to the second-order type specializing “Quality Universal”, we may define a second-order type specializing “Substantial Universal” whose instances are characterized by instances of the former. In the example at hand, we may define the domain second-order type “Physical Object Type by Color” with instances “Blue Object” and “Green Object”, each of which is characterized by an instance of “Color Type”. This is captured in the domain conceptual model of Fig. 19, in two ways. First, the characterization relation between “Physical Object” and “Color” is specialized to link each instance of “Physical Object Type by Color” to the corresponding instance of “Color Type” (e.g., every instance of “Blue Object” bears an instance of “Blue-Toned Color”, every instance of “Green Object” bears an instance of “Green-Toned Color”, and so on). Second, a one-to-one association between “Physical Object Type by Color” and “Color Type” is used to represent that each instance of the former is defined considering an instance of the latter.

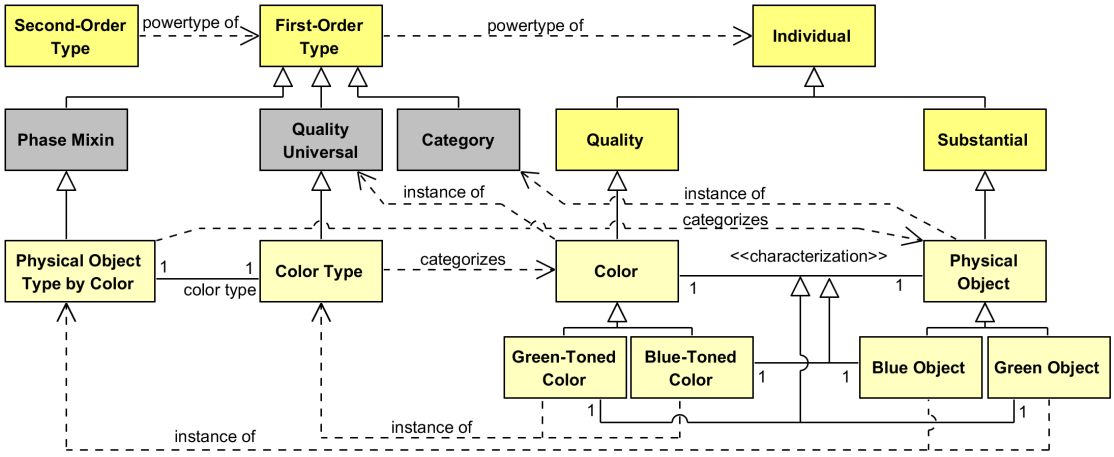

Fig. 19. Making explicit the relation between substantials and qualities taxonomies.

To determine which of leaf ontological category is specialized by the introduced second-order type, we must consider: (i) the mutability of the quality involved in the adopted classification criteria (in this case, the mutability of the color of a physical object) and (ii) the ontological category of its base type (in this case, the ontological category of “Physical Object”).

For a mutable quality, the instances of the second-order type are in general anti-rigid types. For example, considering that the color of a physical object may change during its life cycle (migrating from the extension of a color type to another), we conclude that instances of “Physical Object Type by Color” are anti-rigid types (e.g. an apple may change from green to red, ceasing to be an instance of “Green Object” and becoming an instance of “Red Object”). Considering “Physical Object” a category, two alternatives remain (“Phase” and “Phase Mixin”) for the leaf ontological category being specialized by the introduced second-order type, dependent on whether its instances carry a principle of identity. Considering that instances of “Physical Object Type by Color” have instances of different kinds, we define in Fig. 19 that it specializes “Phase Mixin”.

In the cases in which the qualities involved are immutable, we need to consider further the rigidity of the base type in order to determine whether the instances of the second-order type at hand are rigid or anti-rigid types. For an immutable quality and a rigid base type, the instances of the second-order type are rigid types, and thus, the second-order type must specialize either “Kind”, “Subkind” or “Category”. For example, consider a second-order type “Person Type by Gender” that partitions “Person” using a quality “Gender” as classification criteria. If we consider that the gender of a person is immutable and that “Person” is a rigid type, we can conclude that instances of “Person Type by Gender” (e.g. “Man” and “Woman”) are rigid types. Further, since the base type is a kind, “Person Type by Gender” must specialize “Subkind”. In contrast, for an immutable quality and anti-rigid base type,

the instances of the second-order type are anti-rigid types and thus, the second-order type must specialize either "Phase", "Role", "Phase Mixin" or "Role Mixin". For example, considering that "Child" is a phase, if we define a second-order type "Child Type by Gender" that uses gender as criteria to partition "Child" we have that "Child Type by Gender" specializes "Phase" with instances "Boy" and "Girl".

Specializations of Mode Universal. Similarly to quality universals, mode universals may also be categorized according to some criteria. Thus, we may define a *second-order type* that *specializes* "Mode Universal" and *categorizes* an instance of "Mode Universal" using properties of the mode individuals as classification criteria. For example, we may define a *second-order type* "Disease Type" that *categorizes* "Disease" having instances such as "Diabetes" and "Hemophilia" (see Fig. 20).

Considering that modes are reification of intrinsic properties inhering on substantials, similarly to quality universal, mode universals may be related to substantial universals through characterization. Further, the same criteria used by a second-order type specializing " Mode Universal" to categorize an instance M of "Mode Universal" can be used to categorize an instance S of "Substantial Universal" distinguishing instances according to the subtype of M they bear. For example, consider an instance of "Phase" called "Diseased Person" which encompasses all persons bearing some disease. To capture that every diseased person bears a disease, we may relate "Diseased Person" to "Disease" through characterization. We may define a second-order type "Diseased Person Type" that categorizes "Diseased Person" according to the type of disease, having instances such as "Diabetic Person" and "Hemophiliac Person". In this scenario, each instance of "Diseased Person Type" is characterized by a specific instance of "Disease Type". This fact is captured in Fig. 20 by (i) specializations of the *characterization* between "Diseased Person" and "Disease", and (ii) a one-to-one relationship between "Diseased Person Type" and "Disease Type". Applying the same analysis we conducted in the previous section to determine which leaf ontological category is specialized by the introduced second-order type, we conclude that "Diseased Person Type" specializes "Phase" since the base type "Diseased Person" is a "Phase".

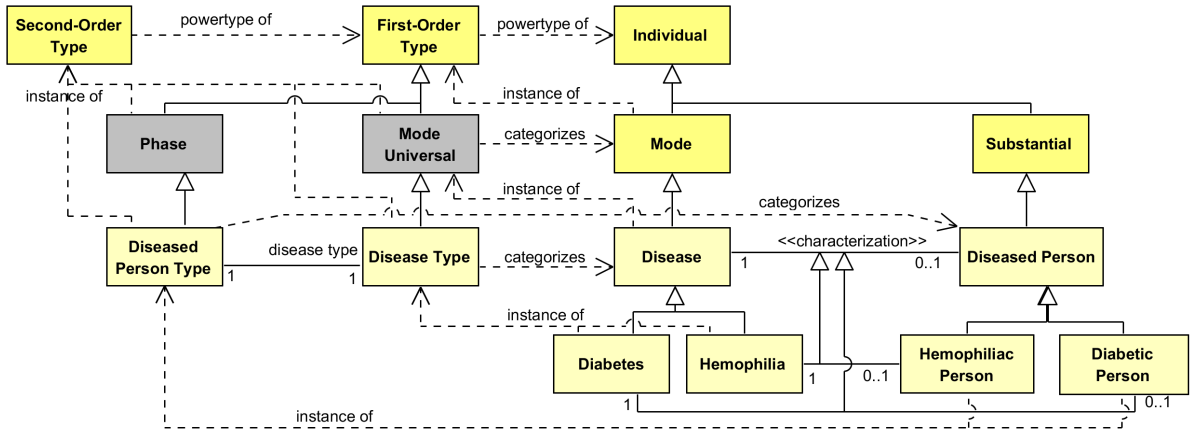

Fig. 20. Second-order types specializing "Mode Universal" and the relationship between modes taxonomies and substantials taxonomies..

Specializations of Relator Universal. As discussed in [18], besides having the power of connecting entities, relators also have their own intrinsic properties. Using these intrinsic properties as classification criteria, we may define specializations of "Relator Universal" that *categorize* instances of "Relator Universal". For example, we may define a second-order type "Employment Type by Duration" that *partitions* "Employment" having "Temporary Employment" and "Tenured Employment" as instances (see Fig. 21).

Relator universals may also be specialized considering properties of the connected entities (the relata). In this case, we may define a second-order type that *specializes* "Relator Universal" and *categorizes* an instance of "Relator Universal" using properties of the relata as criteria. For example, considering that important characteristics of employments (e.g., the types of claims and commitments en-

tailed) vary according to the sort of the organization which is the employer, we may define a second-order type "Employment Type by Employer Nature" that specializes "Relator Universal" and *partitions* "Employment" having instances such as "Public Employment" and "Private Employment". This scenario is illustrated in Fig. 21.

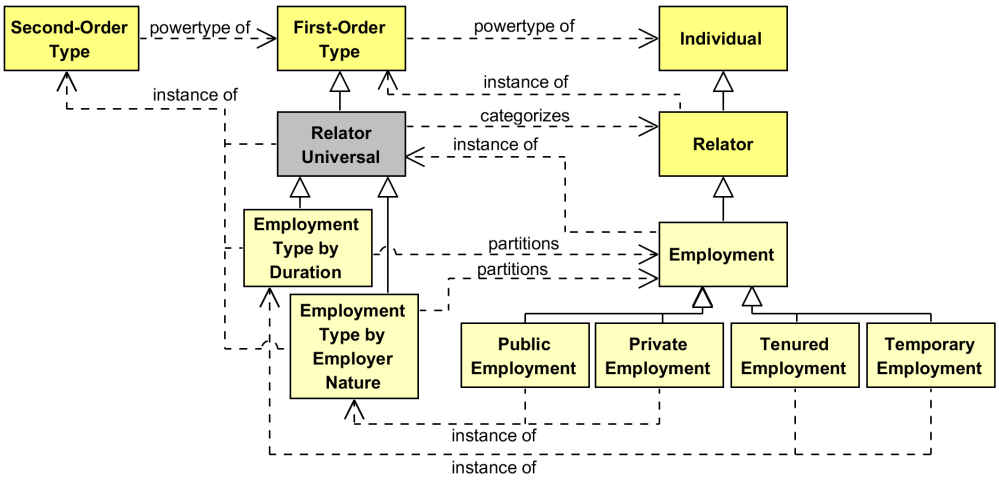

Fig. 21. Second-order types specializing "Relator Universal".

In the previous sub-sections, we discussed that taxonomies of substantials may be created reflecting taxonomies of intrinsic moments. The same reasoning can be applied to taxonomies of relators: the criteria used by a second-order type specializing "Relator Universal" to categorize an instance of "Relator Universal" can be used to categorize an instance of "Role" mediated by the relator universal, distinguishing instances according to the type of relational properties they bear. For example, consider the role "Employee" encompassing all persons having an employment. As we have discussed, to capture that each employee has at least one employment, we may relate "Employee" to "Employment" through mediation. Since every employee has employments, we may define a second-order type "Employee Type" that categorizes "Employee" according to the type of employment, having instances such as "Temporary Employee" and "Tenured Employee". Thus, every instance of "Employee Type" would have as instances objects having employments of a specific "Employment Type by Duration". This fact is captured in the domain conceptual model of Fig. 22 in two ways. First, the mediation relation between "Employee" and "Employment" is specialized to link each instance of "Employee Type" to the corresponding instance of "Employment Type by Duration" (e.g., every instance of "Temporary Employee" bears a relation with at least one instance of "Temporary Employment", every instance of "Tenured Employee" bears a relation with at least one instance of "Tenured Employment", and so on). Second, a one-to-one association between "Employee Type" and "Employment Type by Duration" is used to represent that each instance of the former is defined considering an instance of the latter.

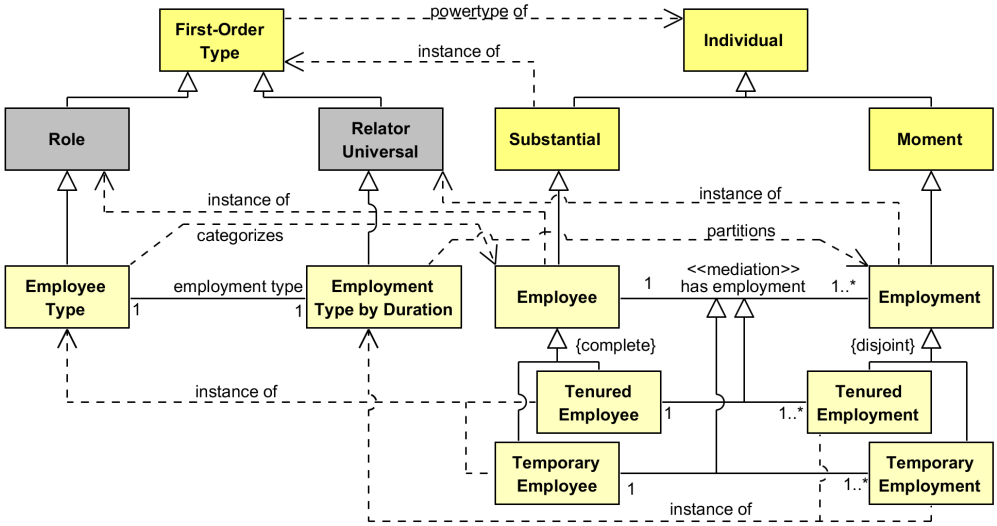

Fig. 22. Explicitly representing the relation between roles and relators taxonomies.

5.4 Summary

Table 2 summarizes the discussion concerning specializations of "Substantial Universal". It presents the admissible ontological categories for base types of second-order types specializing each leaf category of "Substantial Universal". The columns representing admissible base types are marked with a capital 'X' while inadmissible base types are marked with an hyphen '-' (e.g., the table informs that it is admissible for a second-order type specializing "Subkind" to have an instance of "Kind" as base type, but it is inadmissible for a specialization of "Subkind" to have an instance of "Subkind" as base type.

Table 2. Allowed base types for specializations of Substantial Universal

	First-order Base Type						
Second-order Type	Instance of "Kind"	Instance of "Subkind"	Instance of "Role"	Instance of "Phase"	Instance of "Category"	Instance of "Role Mixin"	Instance of "Phase Mixin"
Specialization of "Kind"	-	-	-	-	X	-	-
Specialization of "Subkind"	X	-	-	-	X	-	-
Specialization of "Phase"	X	X	X	X	X	X	X
Specialization of "Role"	X	X	X	X	X	X	X
Specialization of "Category"	-	-	-	-	X	-	-
Specialization of "Phase Mixin"	-	-	-	-	X	X	X
Specialization of "Role Mixin"	-	-	-	-	X	X	X

As discussed in section 5.3, the rules concerning the admissible ontological categories for base types of second-order types specializing "Moment Universal" are simpler, as sortality and rigidity are not addressed here for these types. In summary: (i) a specialization of "Quality Universal" must have an instance of "Quality Universal" as base type; (ii) a specialization of "Mode Universal" must have an instance of "Mode Universal" as base type; and (iii) a specialization of "Relator Universal" must have an instance of "Relator Universal" as base type.

6 Related Work

Two early attempts to address multi-level modeling, namely *power types* [12, 13] and *materialization* [19], raised from the identification of patterns to represent the relationship between a class of catego-

ries and a class of more concrete entities. Despite their different origins, they are based on similar conceptualizations [20] addressing similar concerns. Both approaches establish a relationship between two types such that the instances of one are specializations of another. The power type approach was incorporated in the UML [21], and the language currently includes a *power type association* that relates a classifier (power type) to a generalization set composed by the generalizations that occur between the base classifier and the instances of the power type. Since OntoUML [3] does not add to UML any support for higher-order types, the only multi-level modeling support provided to OntoUML users is UML's support for *power types*. Because of its dependence on the generalization set construct, the UML power type pattern can only be applied when specializations of the base type are explicitly modeled (otherwise there would be no generalization set). We consider this undesirable, as it would rule out simple models that are possible in our approach, e.g., one defining "Dog Breed" as a power type of "Dog", without forcing the modeler to enumerate the instances of "Dog Breed". We capture the fact that the instances of "Dog Breed" form a partition of "Dog", regardless of their representation in a model. This is a more adequate choice considering our focus on representing a conceptualization as accurately as possible; this allows the representation of a conceptualization of dogs and dog breeds in general, without mention of specific dog breeds.

In [22], Erikson et al. propose a UFO-based approach that tries to avoid second-order types by employing a pattern based on the so-called *ontological square* comprising the categories of Substantial Univ./Substantial Individual and Moment Univ./Moment Individual, as well as their mutual relations. They provide an example in which "Horse" is considered a *substantial type,* a horse named "Prancer" is a *substantial object* (instance of "Horse"), "Breed" is a *moment type* and "Shetland Pony" is a *moment object* (instance of "Breed"). Since both *Prancer* and *Shetland Pony* are objects, there is no instance of relation between them. According to the authors, each instance of *Horse* is related to one instance of *Breed* and one instance of *Breed* is related to many instances of *Horse.* Their assumption that the same *moment object* can be related to various *substantial objects* is a misinterpretation of a basic rule of the foundational ontology. In UFO, the relation between *moment objects (individuals)* and *substantial objects (individuals)* is that of *inherence.* In the ontology literature, in general, and in UFO, in particular, it is not possible for an intrinsic moment to inhere in two different individuals. What the authors seem to intend to represent is actually the relation between a *property* (in the ontological sense) "Shetland Pony" and a number of individuals in which this property is *exemplified* (also in the ontological sense [3]). However, under this interpretation, "Shetland Pony" becomes a universal and "Breed" a second-order universal, defeating what they were trying to accomplish with their approach. Besides this ontological problem, the authors ignore the intuitive mechanisms of defining subtypes of a type according to properties of their instances and the benefits of such mechanisms. For example, using such approach, there is no support to represent properties that inheres only in instances of "Shetland Pony".

Atkinson and Kühne have proposed a deep instantiation based approach [10, 20] as a means to provide for multiple levels of classification whereby an element at some level can describe features of elements at each level beneath that level. The notions of *clabjects* and *potency* are central to that approach. Observing that types may be instances of other types, Atkinson and Kühne coined the term *clabject,* emphasizing that every instantiable entity has both a type (or class) facet and an instance (or object) facet which are equally valid [23]. Their approach is based on the idea of assigning to *clabjects* and *fields* (attributes and slots) a *potency* which defines how deep the instantiation chain produced by that *clabject* or *field* may become. When a *clabject* is instantiated from another *clabject* the potencies of the created *clabject* and of its *fields* are given by the original *clabject* and *fields* potencies decremented by one. Objects have potency equal to zero indicating they cannot be instantiated. If the potency of a *field* becomes zero then a value can be assigned to that *field.* For example, we could define a *clabject* "Mobile Phone Model" with an attribute "IMEI" assigning a potency of 2 to both the type and the attribute, meaning that instances of instances of "Mobile Phone Model" are assigned a value to the "IMEI" attribute.

The authors consider that the main benefit of deep instantiation based approach is to reduce "accidental complexity" in domain models since it supports multi-level modeling without the need of introducing what they consider superfluous types (the required base type in the power type pattern) [20]. In the aforementioned example, the concept of "Mobile Phone" could be omitted from the domain model. While the deep instantiation approach can reduce the number of entities represented in a

model, this strategy should be used with parsimony. Important consequences of omitting base types in the current deep instantiation approach are that the modeler becomes unable to express whether the instances of a higher-order type ("Mobile Phone Model" in previous example) are disjoint and/or covering types((i.e., we are unable to distinguish which form of *categorization* would apply). We are also prevented from determining metaproperties (such as e.g., rigidity and sortality) of the base type ("Mobile Phone" in previous example). Further, as discussed in [3], conceptual models should always include kinds that define the principle of identity of individuals (in the example this type is mobile phone). If these types are omitted (and incorporated into higher-order types by using the notion of potency), the source of the principle of identity becomes hidden. Note that the patterns that we have defined in section 5 address these issues specifically. For example, one could define "Mobile Phone Model" as a second-order type that specializes "Subkind" and partitions the first-order type "Mobile Phone" meaning that a mobile phone cannot instantiate two models (given the semantics of the partitioning relation) and that each mobile phone is an instance of the every same model throughout its existence (since instances of "Mobile Phone Model" are also instances of "Subkind"). It is worth noticing that the deep instantiation approach allows the modeler to represent the base type if it is deemed desirable. However, the approach does not provide constructs to represent the relation between the base type and the higher-order type, not distinguishing thus between the different possible kinds of cross-level relations. Moreover, in [10, 20] the authors are not aimed at discussing issues concerning ontological foundations for types, which are key subjects here.

Some recent deep instantiation approaches [24] include the notions of attribute durability and mutability. The durability of an attribute indicates how far the attribute spans in an instantiation tree. The mutability of an attribute defines how often the attribute value can be changed over the instantiation tree. As we have discussed in section 5, it is possible to define second-order types using properties inherent to instances of the base type as criteria to distinguish between instances. Using the notions of durability and mutability it is possible to capture, in deep modeling, a particular kind of this relation between second-order types and attributes of base types. Consider, for example, a second-order type "Physical Object Type by Color" that *categorizes* "Physical Object" according to the color inhering in the individuals. Consider also that we represent the colors of physical objects just as nominal values, e.g. blue, red, green and so on. In deep modeling, we could capture this scenario by defining "Physical Object Type by Color" as a clabject with potency 2 having an attribute "color" with durability 2 and mutability 1. This way the attribute "color" will be given a value at the first instantiation of "Physical Object Type by Color" (e.g., stating that the "Blue Object" has a blue "color"), and that such value will determine the value of "color" for the instances of instances of "Physical Object Type by Color" (thus, all instances of "Blue Physical Object" have blue as value for "color").

This is a useful language mechanism to capture constraints between types in adjacent levels, however, this representation strategy is only capable of capturing the constraints in which an upper level type defines the exact value that must be assigned to attributes in the lower order. It is not applicable, for example, to constraints considering possible regions of values assignable to an attribute. For example, in section 5.3 we consider that each instance of "Physical Object Type by Color" defines a certain region in the color domain such that their instances must have a color inside the defined region. In this case, each instance of "Physical Object Type by Color" does not define an exact value for the color attribute of its instance but a set of possible values. For example, "Blue Object" is not defined as a type whose instances have "blue" as value for color but a type whose instances have a blue-toned color, i.e. a color in a specific region of the color domain. This region of the domain could the captured, for example, as ranges of hue, brightness and saturation. Thus, since in this scenario the instances of "Physical Object Type by Color" do not define the exact value for the color attribute of their instances, it could not be captured applying the notions of durability and mutability.

The *multi-level objects* (or *m-objects)* [9] is another multi-level modeling approach that applies the notion of *deep instantiation*. This approach is based on the notion of objects that "encapsulate different levels of abstractions that relate to a single domain concept", the so-called *m-objects*. Considering the *mobile phone* example, the modeler could define an m-object named "Mobile Phone Model" with two levels of abstraction, namely *type* and *physical entity* levels. Properties that are characteristics of a mobile phone model, such as *screen size*, should be defined at the *type level* while the ones exemplified by mobile phones, such as *IMEI*, should be defined at the *physical entity level*. The approach defines a *concretize* relationship to associate different m-objects. For example, applying the *concretize*

relation to "Mobile Phone Model" we could create an m-object named "IPhone6" attributing values to the properties defined at the type level of "Mobile Phone Model" and at the same time "inheriting" the properties defined at the physical entity level of "Mobile Phone Model" such as "IMEI". Thus, the *concretize* relationship semantically overloads instantiation and specialization. Given that these are relations of different ontological nature, we believe this could affect the understandability and usability of the approach. Similarly to Atkinson and Kühne's proposal [10, 20] the approach leads to a model with fewer elements, but prevents us from expressing important aspects of the first- and second-order types.

Finally, the strategy previously used in OntoUML [3] was one in which the types represented in conceptual models could only instantiate the universals in UFO's taxonomy of universals. These were represented by a fixed set of UML stereotypes, and thus a conceptual model could only have first-order types. In that approach, the axioms of the foundational ontology had to be incorporated into the syntax and semantics of the language profile (e.g., translated into corresponding syntactic rules as shown in [3], or incorporated in a transformation of OntoUML into a logical formalism). This additional step is not necessary here as the structural relations and axioms of UFO-MLT are directly incorporated in the domain ontology. For example, concerning the combinations of the specialization patterns for second-order types, it is inadmissible for a domain second-order type that specializes "Kind" and "Subkind" to be subordinate to a domain second-order type which specializes "Role" or "Phase". This is a consequence of the constraint in UFO that rules out the specialization of an anti-rigid universal by a rigid universal, together with the definition of subordination in MLT.

7 Conclusions and Future Work

In this paper we have extended the Unified Foundational Ontology with the MLT multi-level theory in order to provide foundations for ontology-based multi-level conceptual modeling. MLT is founded on the notion of (ontological) instantiation, which is applied regularly across levels ("orders"). An important basic pattern of the theory has influenced significantly our approach: types *instantiate* a type at an immediately higher order and *specialize* the basic type of the order to which they belong.

We have shown how the elements of MLT can be used to serve as the topmost layer of a hierarchy of conceptual models, from a foundational ontology to conceptual domain models. The concepts of the foundational ontology *instantiate* and *specialize* elements of MLT, respecting its axioms and using structural relations and patterns of MLT. In turn, the concepts of the conceptual domain model *instantiate* and *specialize* UFO-MLT, respecting MLT and UFO axioms and patterns. The result is an approach to define conceptual domain models that can represent types as well as types of types while adhering to the rules of a foundational ontology.

UFO's original taxonomy of (first-order) universals is leveraged in order to provide patterns for types of types in the domain model. These patterns guide the modeler in the definition of higher-order types and their relations allowing the modeler to express modal properties of instances of higher-order types. To the best of our knowledge, this is the first initiative on identifying patterns and constraints for higher-order types based on a foundational ontology. Here we extend [14] providing fuller treatment for specializations of "Sortal Universal" and also dealing with specializations of "Mixin Universal" and "Moment Universal". These extensions to the original proposal bring important benefits to the UFO-MLT approach.

Firstly, let us elaborate on the importance of also considering the case of mixin universals. While sortal universals constitute the bulk of a domain conceptual model representing the most recurrent types in these models, mixin universals play a complementary role, classifying entities that "cross the boundaries" of multiple *kinds*. These universals are perceived to be of fundamental importance in structuring the specification of conceptual models [25, 26], particularly, in the case of large taxonomic structures [27]. For instance, a mixin universal such as "Insured Item" (that classifies entities of different *kinds* such as buildings, paintings, cars and body parts) provides for a model refactoring mechanism that captures in a concise and elegant manner the common properties shared by the instances of these multiple kinds (e.g., the relational property of being associated via an insurance policy with an insurer). Furthermore, as demonstrated in [28], there are certain recurrent conceptual modeling prob-

lems (e.g., the problem of modeling *roles with disjoint allowed types*) that can hardly be solved in a satisfactory manner without the use of mixin universals.

Secondly, introducing in our approach the explicit representation of moment universals allows us to deal with recurrent situations in ontology-driven conceptual modeling when moment individuals are *reified* (objectified). This includes, as discussed in [29], the representation of intrinsic moments (such as "Paul's Dengue Fever" or "Clara's knowledge of Dutch") that are the ontological counterpart of *Weak Entities* in traditional conceptual modeling. Their explicit representation brings many benefits to the practice of conceptual modeling, which include addressing the so-called *"Counting Problem"* [30], the representation of attributes that can be projected into multiple alternative attribute value spaces (datatypes) [29],the representation of change and modal properties of particular *aspects* of things [31], and the support for temporal reasoning with conceptual models [32]. Further, there are several classical conceptual modeling problems in the modeling of *relational properties* that cannot be solved in a satisfactory manner without the reification of these properties. These include the problem of *collapsing single-tuple and multiple-tuple cardinality constraints* [33], the problem of modeling particular type taxonomic structures involving relational properties [34] and, again, the problem of modeling change and modality of relationships [18].

Another consequence of employing MLT concerns the engineering of UFO itself. UFO's taxonomies can now be explained in terms of instantiation of higher-order types. Further, as shown in section 4 the relations of MLT (such as categorization) can be used to explain how elements in UFO's taxonomy of universals relate to elements in the taxonomy of individuals. Further, while we have focused on the definition of domain models, the approach discussed here forms the basis for further extension of UFO itself, as well as to include core ontologies in the hierarchy of models between the foundational ontology and domain models. We will apply this approach to improve the formalization of UFO-based ontologies whose conceptualizations span multiple levels of classifications (e.g., the UFO-S core ontology for services [35] and the O3 organizational ontology [36]).

Finally, we should stress that it is not our intention in this paper to propose a multi-level language, and that our use of a notation inspired in UML has been solely illustrative. As discussed in [3], a reference ontology can be used to inform the revision and redesign of a modeling language, both through the identification of semantic overload, construct deficit, construct excess and construct redundancy, and through the definition of modeling patterns and semantically-motivated syntactic constraints. In [37], we have used MLT to revise the UML powertype support, demonstrating that the current support lacks *expressivity*, *clarity*, and *parsimony*. This has resulted in a UML extension addressing the identified limitations. A natural application for UFO-MLT is to promote the redesign of an ontologically well-founded modeling language such as OntoUML with support for multi-level modeling.

Acknowledgements. This research is funded by the Brazilian Research Funding Agencies CNPq (grants number 311313/2014-0, 485368/2013-7 and 461777/2014-2) and CAPES/CNPq (402991/2012-5). Victorio A. Carvalho is funded by CAPES.

References

1. Mylopoulos, J.: Conceptual Modeling and Telos. In: Loucopoulos, P., Zicari, R. (Eds.), Conceptual modeling, databases and CASE, pp. 49-68, Wiley (1992)
2. Olivé, A.: Conceptual Modeling of Information Systems. Springer (2007)
3. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models. University of Twente, Enschede, The Netherlands (2005)
4. Wand, Y., Weber, R.: An ontological evaluation of systems analysis and design methods. In: Falkenberg E., Lingreen, P. (Eds.), Information System Concepts: An In-Depth Analysis. Elsevier Science Publishers B.V., North-Holland (1989)
5. Wand, Y. and Weber, R.: On the ontological expressiveness of information systems analysis and design grammars. Journal of Information Systems, (3), 217-237 (1993)
6. Guarino, N.: Formal Ontology and Information Systems. In Guarino, N. (ed.), Formal Ontology in Information Systems, Proc. FOIS 1998. IOS Press, Amsterdam: pp. 3-15 (1998)
7. Guizzardi, G., Wagner, G., Almeida, J.P.A., Guizzardi, R.S.S.: Towards Ontological Foundation for Conceptual Modeling: The Unified Foundational Ontology (UFO) Story. Applied Ontology, Vol. 10, issues 3-4, IOS Press (2015)
8. U.S. Department of Defense: Data Modeling Guide (DMG) for an Enterprise Logical Data Model (ELDM), available online: http://www.omgwiki.org/architecture- ecosystem/lib/exe/fetch.php?media=dmg_for_enterprise_ldm_v2_3.pdf.

9. Neumayr, B., Grün, K., Schrefl, M.: Multi-level domain modeling with m-objects and m-relationships. In: Proc. 6th Asia-Pacific Conf. on Conceptual Modeling, New Zealand (2009)
10. Atkinson, C., Kühne, T.: The Essence of Multilevel Modeling. In: Proc. Of the 4th International Conference on the Unified Modeling Language. Toronto, Canada (2001)
11. Carvalho, V. A., Almeida, J. P. A.: Toward a well-founded theory for multi-level conceptual modeling. . Software & Systems Modeling, Springer Berlin Heidelberg (2016)
12. Cardelli, L.: Structural Subtyping and the Notion of Power Type. In Proc. Of the 15th ACM Symposium of Principles of Programming Languages, pp. 70-79 (1988)
13. Odell, J.: Power types. In: Journal of Object-Oriented Programing, 7(2), pp. 8-12.(1994)
14. Carvalho, V.A., Almeida, J.P.A., Fonseca, C.M., Guizzardi, G.: Extending the Foundations of Ontology-based Conceptual Modeling with a Multi-Level Theory. In: 34rd International Conference on Conceptual Modeling (ER2015) (2015)
15. Guarino, N., Welty, C.: Identity and Subsumption. In Green, R., Bean, C. A., Hyon Myaeng, S. (eds.), The Semantics of Relationships: An Interdisciplinary Perspective. Kluwer, pp.111-126 (2002)
16. Armstrong, D. M.: Universals as attributes. In J. Kim & E. Sosa (Eds.), Metaphysics: An anthology, pp. 198–208, Oxford: Blackwell (1999)
17. Gärdenfors, P.: Conceptual Spaces: the Geometry of Thought. MIT Press, Cambridge, USA (2000)
18. Guarino, N., Guizzardi, G.: We need to discuss the relationship: revisiting relationships as modeling constructs. In 27th International Conference on Advance Information Systems Engineering (CAISE 2015), Stockholm, Sweden (2015)
19. Pirotte, A., Zimanyi, E., Massart, D., Yakusheva, T.: Materialization: a powerful and ubiquitous abstraction pattern. Proc. 20th Int. Conf. Very Large DataBases, pp. 630–641 (1994)
20. Atkinson, C., Kühne, T.: Reducing accidental complexity in domain models. Software & Systems. Modeling, 7(3), pp 345-359. Springer-Verlag (2008)
21. OMG : UML Superstructure Specification – Version 2.4.1 (2011)
22. Eriksson. O., Henderson-Sellers, B., Ågerfalk, P. J.: Ontological and linguistic metamodeling revisited: A language use approach. Information and Software Technology, 55(12), pp. 2099-2124. Elsevier (2013)
23. Atkinson, C., Kühne, T.: Meta-level Independent Modeling. In International Workshop "Model Engineering" (in conjunction with ECOOP'2000), Cannes, France (2000)
24. Kennel, B.: A Unified Framework for Multi-Level Modeling. University of Mannheim (2012)
25. Jacobson, I., Booch, G., Rumbaugh, J.: The Unified Software Development Process. Addison-Wesley (1998)
26. Booch, G.: Object-Oriented Analysis and Design. Benjamin-Cummings (1994)
27. Welty, C., Guarino, N.: Supporting Ontological Analysis of Taxonomic Relationships. Data and Knowledge Engineering, 39(1), pp. 51-74 (2001)
28. Guizzardi, G., Wagner, G., Guarino, N., van Sinderen, M.:An Ontologically Well-Founded Profile for UML Conceptual ‚Models. In Proc. 16th International Conference on Advances in Information Systems Engineering (CAiSE). Springer-Verlag , Berlin (2004)
29. Guizzardi, G., Masolo, C., Borgo, S.: In the Defense of a Trope-Based Ontology for Conceptual Modeling: An Example with the Foundations of Attributes, Weak Entities and Datatypes. In Proc. of the 25th International Conference on Conceptual Modeling (ER'2006), Arizona, USA (2006)
30. Guizzardi, G.: Agent Roles, Qua Individuals and The Counting Problem. Invited Chapter in Software Engineering of Multi-Agent Systems, vol. IV, P. Giorgini, A.Garcia, C. Lucena, R. Choren (eds.), Springer-Verlag (2006).
31. Guizzardi, G., Guarino, G., Almeida, J.P.A.: Ontological Considerations about the Representation of Events and Endurants in Business Models. In Proc. of the 14th International Conference on Business Process Management (BPM'16). Rio de Janeiro, Brazil (2016)
32. Guizzardi, G., Zamborlini, V.: Using a Trope-Based Foundational Ontology for Bridging different areas of concern in Ontology-Driven Conceptual Modeling. Science of Computer Programming, Elsevier (2014)
33. Guizzardi, G., Wagner, G.: What's in a Relationship: An Ontological Analysis. In Proc. of the 27th International Conference on Conceptual Modeling (ER'2008). Barcelona, Spain (2008)
34. Costal, D., Goméz, C., Guizzardi, G.: Formal Semantics and Ontological Analysis for Under-standing Subsetting, Specialization and Redefinition of Associations in UML. In Proc of the 30th International Conference on Conceptual Modeling (ER 2011). Brussels, Belgium (2011)
35. Nardi, J. C., Falbo, R., Almeida, J. P. A., Guizzardi, G., et al.: A commitment-based reference ontology for services. Information Systems, 51, p.1 (2015)
36. Pereira, D., Almeida, J. P. A.: Representing Organizational structures in an enterprise architecture language. In 6th Workshop on Formal Ontologies meet Industry (2014)
37. Carvalho, V.A., Almeida, J.P.A., Guizzardi, G.: Using a Well-Founded Multi-Level Theory to Support the Analysis and Representation of the Powertype Pattern in Conceptual Modeling. In: 28th Intl. Conf. on Advanced Information Systems Engineering (2016)