Towards a Well-Founded Theory for Multi-Level Conceptual Modelling

Victorio A. de Carvalho[1,2] and João Paulo A. Almeida[1]

[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
victorio@ifes.edu.br; jpalmeida@ieee.org

Abstract. Multi-level conceptual modelling addresses the representation of subject domains dealing explicitly with multiple classification levels. Despite the recent advances in multi-level modelling techniques, we believe that many challenges in multi-level conceptual modelling still stem from the lack of theory that: (i) formally characterizes the nature of classification levels, and (ii) precisely defines the structural relations that may occur between elements of different classification levels. This work aims to fill this gap by proposing an axiomatic theory that can be considered a reference top-level ontology for types in multi-level conceptual modelling. The theory provides the modeller with basic concepts and patterns to articulate domains that require multiple levels of classification as well as to inform the development of well-founded languages for multi-level conceptual modelling. The theory includes intra-level structural relations that are used to define expressive multi-level models and cross-level relations that allow us to account for and incorporate the different notions of power type in the literature. Since all the relations in the theory are ultimately explained in terms of instantiation between entities of adjacent levels, it is capable to harmonize power type and clabject-based approaches.

Keywords: Multi-level modelling, conceptual modelling, power types, clabjects, ontology

1 Introduction

Conceptual modelling is the activity of formally describing some aspects of the physical and social world around us for the purposes of understanding and communication [19]. It is generally considered a fundamental activity in information systems engineering [24], in which a given subject domain is described independently of specific implementation choices [14]. The main artefact of this activity is a *conceptual model*, i.e., a specification aiming at representing a conceptualization of the subject domain of interest. Conceptual models are often used as a basis for the construction and evolution of information systems, which justifies the interest in the activity of conceptual modelling from the perspective of information systems engineering [24].

Given the scope and purpose of conceptual modelling, suitable techniques for this endeavour should be based on abstractions with consideration for human cognition and common sense [14, 13]. With this respect, there is ample psychological evidence to support the hypothesis that humans conceive of the physical and social world using some notion of "categories" and use categorization or classification strategies since a pre-language age of 3-4 months (see [14], pp. 114-118). Thus, it is no surprise that a vast majority of conceptual modelling techniques are based on notions such as "class" and "type", and that subject matter experts often refer to "kinds", "categories" and "sorts" in their accounts of a subject domain.

In several subject domains, the categorization scheme itself is part of the subject matter, and thus experts make use of categories of categories in their accounts. For instance, considering the software development domain [11], project managers often need to plan according to the *types of tasks* to be executed during the software development project (e.g. "requirements specification", "coding"). They may also need to classify those *types of tasks* giving rise to *types of types of tasks*. In this case, "requirements specification" and "coding" could be considered as examples of "technical task types", as opposed to "management task types". Finally, during project development, they need to track the execution of individual tasks (e.g. specifying the requirements of the system X). Thus, to describe the conceptualization underlying the software development domain, one needs to represent entities of different (but nonetheless related) classification levels, such as *tasks (specific individual occurrences), types of tasks*, and *types of types of tasks*. Other examples of multiple classification levels come from domains such as that of organizational roles (or professional positions) [26], biological taxonomy [18] and artefact types (e.g., product types) [21].

The need to support the representation of subject domains dealing with multiple classification levels has given rise to what has been referred to as multi-level modelling [4, 21]. Techniques for multi-level conceptual modelling must provide modelling concepts to deal with types in various classification levels and the relations that may occur between those types.

The *power type* pattern [6, 23] is an example of an early approach for multi-level modelling and is used to model situations in which the instances of a type (the *power type)* are specializations of a lower-level type (the *base type)*. While some prominent approaches for multi-level modelling are based on the notion of *power type*, there is no consensus about the exact definition of a *power type*[1] and most approaches based on the notion lack a formal account for it. Further, most of the power type based approaches represent the relation between a power type and a base type as a regular association with no specialized semantics.

Other prominent approaches for multi-level modelling (such as [3]) propose to treat the instantiation between arbitrary adjacent levels uniformly [1], i.e., they defend that the instantiation relations between specific individuals and their types should also be applied to the instantiation relation occurring between types of adjacent classification levels. To meet this challenge it is necessary to admit the existence of entities, which are, simultaneously, type (class) and instance (object). The authors have coined the term "clabject" to emphasize this dual facet of classes in a generalized multi-level scheme.

Despite the recent advances in multi-level modelling techniques, we believe that the literature would benefit from a theory that: (i) formally characterizes the nature of classification levels, and (ii) precisely defines the relations that may occur between elements of different classification levels. Such a theory should be useful to guide the development of well-founded languages

[1] An issue that shall be discussed in this paper in section 4.

for multi-level conceptual modelling and to provide the modeller with basic concepts and patterns to conceptualize domains that require multiple levels of classification.

This work aims to fill this gap by proposing a well-founded theory for multi-level conceptual modelling. The theory is built up from a basic *instantiation* relation and characterizes the concepts of *individuals* and *types*, with types organized in levels related by instantiation. The theory accounts for the notion of power type with two contributions: (i) it clarifies and positions conflicting definitions of power type, and (ii) defines new structural relations for variants of the power type pattern enriching the expressivity of multi-level modelling primitives. The basic entities in the theory and all proposed relations between entities are formally defined through axiomatization in first-order logic.

The resulting theory can be considered a reference top-level ontology for types in multi-level conceptual modelling. Although we do not propose a language for multi-level conceptual modelling, we explore patterns that emerge from the application of the theory as well as modelling constraints to ensure that multi-level models respect the theory axioms. Since our focus is on conceptual modelling (and not language engineering or language metamodelling) we focus our account on what is called "ontological instantiation" in [2] and we are thus unconcerned with "linguistic instantiation".

This paper is further organized as follows. Section 2 presents the basic entities in the theory. Section 3 discusses intra-level relations including specialization and a novel relation we call *subordination*. Section 4 discusses cross-level relations which are the basis to incorporate the notion of *power type* and its variations. Section 5 illustrates the application of the theory to the domain of biological taxonomy. Section 6 discusses related work and section 7 presents conclusions and future steps.

2 Theory Foundations: Basic Types and the Instantiation Relation

The notions of *type* and *individual* are central for our multi-level modelling theory. *Types* are predicative entities that can possibly be applied to a multitude of entities (including types themselves). Particular entities, which are not types, are considered *individuals*.

Each type is characterized by a *principle of application* with which we judge whether the type applies to an entity (e.g., whether something is a Person, a Dog, a Chair) (following [14]). If the principle of application of a type *t* applies to an entity *e* then it is said that *e is an instance of t*. Thus, the *instance of* relation (or *instantiation* relation[2]) maps a type to the entities that fall under the type. The set of instances of a type is called the *extension* of the type [15].

Our multi-level modelling theory is formalized in first-order logic, quantifying over all possible individuals and types. The *instantiation relation* is formally represented by a binary

[2] We are aware that certain approaches such as RM-ODP distinguish the terms *instantiation* and *instance*, but this distinction is not required here, and hence we use the terms interchangeably.

predicate *iof(e,t)* that holds if an entity *e* is instance of an entity *t* (denoting a type). For instance, the proposition *iof(Vitória,City)* denotes the fact that "Vitória" is an instance of the type "City". Further, we consider that there are no relevant types with a trivially false principle of application (hence, the principle of application will always apply to some possible instance).

We build up the theory defining the conditions for entities to be considered *individuals*, with the constant "Individual" in axiom A1. An entity is an instance of "Individual" iff it is not possibly related to another entity through instantiation.

$$\forall x\, \mathrm{iof}(x, \mathrm{Individual}) \leftrightarrow \nexists y\, \mathrm{iof}(y, x) \qquad \text{(A1)}$$

As a multi-level modelling theory, we deal with *types* that have *individuals* as instances as well as with types whose extension is composed by other types. In order to accommodate these varieties of types, the notion of *type order* is used. This notion is based on the ramified hierarchy introduced by Russel to deal with the apparent circularity of impredicative definitions [8]. Types whose instances are individuals are called *first-order types*. Types whose instances are *first-order types* are called *second-order types*. Those types whose extensions are composed by *second-order types* are called *third-order types*, and so on. We use the term *higher-order type* to refer to types with order higher than one.

Axiom A2 characterizes "First-Order-Type" (or shortly "1stOT"), defining a first-order type as an entity whose instances are instances of "Individual". Analogously, A3 and A4 characterize "Second-Order Type" (or "2ndOT") and "Third-Order Type" ("3rdOT"). A3 defines that an entity *t* is a second-order type iff all its instances are first-order types (i.e., instances of "1stOT"), and A4 defines that an entity *t* is a third-order type iff all its instances are second-order types (i.e., instances of "2ndOT").[3]

$$\forall t\, \mathrm{iof}(t, \mathrm{1stOT}) \leftrightarrow (\forall x\, \mathrm{iof}(x, t) \rightarrow \mathrm{iof}(x, \mathrm{Individual})) \qquad \text{(A2)}$$

$$\forall t\, \mathrm{iof}(t, \mathrm{2ndOT}) \leftrightarrow (\forall t'\, \mathrm{iof}(t', t) \rightarrow \mathrm{iof}(t', \mathrm{1stOT})) \qquad \text{(A3)}$$

$$\forall t\, \mathrm{iof}(t, \mathrm{3rdOT}) \leftrightarrow (\forall t'\, \mathrm{iof}(t', t) \rightarrow \mathrm{iof}(t', \mathrm{2ndOT})) \qquad \text{(A4)}$$

Substituting *t* by *Individual* in axiom A2, one can see that "Individual" is an instance of "1stOT" (theorem T1). Analogously, using axioms A3 and A4 we can show that "1stOT" is instance of "2ndOT" and "2ndOT" is instance of "3rdOT" (see theorems T2 and T3).

$$\mathrm{iof}(\mathrm{Individual}, \mathrm{1stOT}) \quad \text{(T1)}$$

$$\mathrm{iof}(\mathrm{1stOT}, \mathrm{2ndOT}) \quad \text{(T2)}$$

$$\mathrm{iof}(\mathrm{2ndOT}, \mathrm{3rdOT}) \quad \text{(T3)}$$

[3] This scheme can be extended to consider as many orders as necessary. Since we have not found real-world examples of types in conceptual modelling with order higher than three, we present our theory here considering only *first-order*, *second-order and third-order types*. Later we discuss how this scheme could be modified to consider an infinite number of orders. However, this modification is not required in our approach.

Axiom A5 states that each entity in our domain of enquiry is necessarily an instance of "Individual", "1stOT", "2ndOT" or "3rdOT" (except "3rdOT" whose type is outside the scope of the formalization). Further, axiom A6 defines that "Individual", "1stOT", "2ndOT" and "3rdOT" have no instances in common (i.e., their extensions are disjoint). This makes the set of extensions of "Individual", "1stOT", "2ndOT" and "3rdOT" a partition of the set of entities considered in the theory (and their union the domain of quantification).

$$\forall x \left(\mathrm{iof}(x, \mathrm{Individual}) \vee \mathrm{iof}(x, \mathrm{1stOT}) \vee \mathrm{iof}(x, \mathrm{2ndOT}) \vee \mathrm{iof}(x, \mathrm{3rdOT})\right) \vee (x = \mathrm{3rdOT}) \qquad \text{(A5)}$$

$$\nexists x \left(\mathrm{iof}(x, \mathrm{Individual}) \wedge \mathrm{iof}(x, \mathrm{1stOT})\right) \vee \left(\mathrm{iof}(x, \mathrm{Individual}) \wedge \mathrm{iof}(x, \mathrm{2ndOT})\right) \vee (\mathrm{iof}(x, \mathrm{Individual}) \wedge \mathrm{iof}(x, \mathrm{3rdOT})) \vee (\mathrm{iof}(x, \mathrm{1stOT}) \wedge \mathrm{iof}(x, \mathrm{2ndOT})) \vee (\mathrm{iof}(x, \mathrm{1stOT}) \wedge \mathrm{iof}(x, \mathrm{3rdOT})) \vee (\mathrm{iof}(x, \mathrm{2ndOT}) \wedge \mathrm{iof}(x, \mathrm{3rdOT})) \qquad \text{(A6)}$$

Axiom A7 defines that two types are equal iff the set of all their possible instances is the same[4]. (Note that this definition of equality only applies to elements which are not *individuals*, hence the 'guard' conditions on the left-hand side of the implication.)

$$\forall t1, t2 \left(\neg \mathrm{iof}(t1, \mathrm{Individual}) \wedge \neg \mathrm{iof}(t2, \mathrm{Individual})\right) \rightarrow$$

$$((t1 = t2) \leftrightarrow (\forall e \ \mathrm{iof}(e, t1) \leftrightarrow \mathrm{iof}(e, t2))) \quad \text{(A7)}$$

Since the *instantiation* relation denotes that an element is a member of the extension of a type, it must be irreflexive, asymmetric and intransitive [15, 17]. Further, *instantiation* relations hold between two elements such that the last is one order higher than the former. In our theory, all these properties are guaranteed by axioms A1 – A7.

Fig. 1 illustrates the elements that form the basis for our multi-level modelling theory, using a notation that is largely inspired in UML. We use the UML class notation to represent the *basic types of the theory* ("Individual", "1stOT", "2ndOT" and "3rdOT"). We use associations as usual to represent relations between instances of the related types. The multiplicity of the associations reflect the constraints in the formalization. For example, each instance of "Individual" is *instance of* at least one instance of "1stOT", and, on the inverse direction, each instance of "1stOT" has at least one instance of "Individual" in its extension. Since UML does not allow for the representation of links between classes, we use dashed arrows to represent relations that hold between the types, with labels to denote the names of the predicates that apply. For instance, a dashed arrow labelled *iof* between "Individual" and "1stOT" represents that the former is an instance of the latter (i.e., that *iof(Individual,1sOT)* holds). In Fig. 1 the dashed arrows are justified by theorems T1-T3. The notation used to elaborate Fig. 1 is used in all further diagrams in this paper.

[4] Note that this is only true for types that apply necessarily to their instances (the so-called rigid types in [12]), so, dynamic classification is not considered in the present theory.

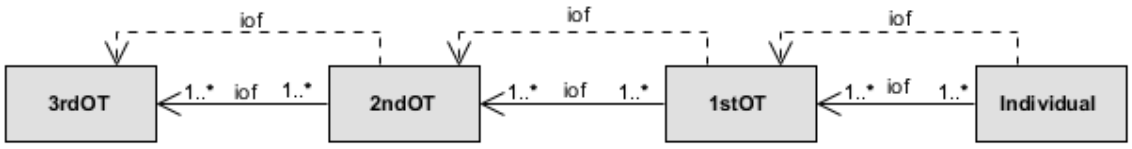

Fig. 1. Basic foundations of our multi-level modelling theory: *basic types* and *instance of* relations.

3 Intra-level structural relations

In this section, we discuss the relations that occur between types of the same order (the intra-level structural relations). All definitions are based on the instantiation relation.

We start with the ordinary specialization between types. Axiom A8 defines that *t1 specializes t2* iff all instances of *t1* are also instances of *t2*. Since instances of "Individual" do not have instances (A1), A8 states that specialization only applies to elements that are not *individuals*. As discussed in [15, 17] *specialization* is a partial order relation (i.e., a reflexive, transitive and antisymmetric relation), which is guaranteed in this theory.

$$\forall t1, t2\ \text{specializes}(t1, t2) \leftrightarrow (\neg \text{iof}(t1, \text{Individual}) \wedge \neg \text{iof}(t2, \text{Individual}) \wedge (\forall e\ \text{iof}(e, t1) \rightarrow \text{iof}(e, t2))) \quad \text{(A8)}$$

According to axiom A8, every type specializes itself. Since this may be undesired in some contexts, we define the *proper specialization* relation (we used the qualifier 'proper' as in 'proper subset' considering that the extension of the specialized type is a proper subset of the extension of the general type [15]). Axiom A9 thus defines that *t1 proper specializes t2* iff *t1 specializes* t2 and is different from it.

$$\forall\, t1, t2\ \ \text{properSpecializes}(t1, t2) \leftrightarrow (\text{specializes}(t1, t2) \wedge t1 \neq t2) \quad \text{(A9)}$$

Insofar as the instances of a type are defined by their *principle of application*, the *proper specialization* relation reflects the fact that the *principle of application* of the specializing type keeps the constraints stated by the *principle of application* of the specialized type and adds some other constraint(s) to it.

Fig. 2 augments Fig. 1 by including the representation of specialization and proper specialization relations. Note that the axioms presented thus far guarantee that these relations may only hold between types of the same order, which is reflected in the diagram.

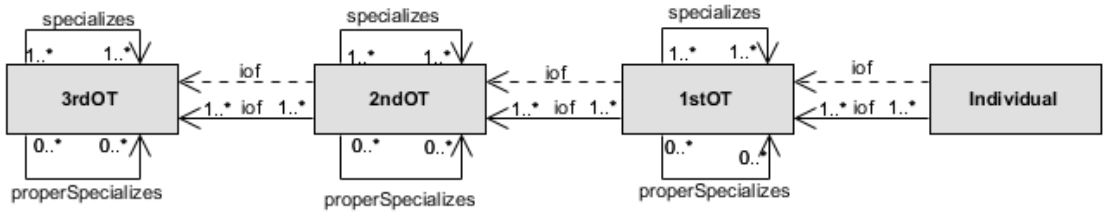

Fig. 2. Intra- level structural relations: specializations and proper specializations.

An important consequence of the axioms presented so far is that any instance of a higher-order type (any instance of "1stOT", "2ndOT", and "3rdOT") specializes the basic type at an order immediately lower order. Thus, every instance of "1stOT" *specializes "*Individual" (theorem T4, following from A8 and A2), every instance of "2ndOT" *specializes* "1stOT" (theorem T5, following from A8 and A3), and so on.

$$\forall t\ \mathrm{iof}(t, \mathrm{1stOT}) \leftrightarrow \mathrm{specializes}(t, \mathrm{Individual}) \quad \text{(T4)}$$

$$\forall t\ \mathrm{iof}(t, \mathrm{2ndOT}) \leftrightarrow \mathrm{specializes}(t, \mathrm{1stOT}) \quad \text{(T5)}$$

$$\forall t\ \mathrm{iof}(t, \mathrm{3rdOT}) \leftrightarrow \mathrm{specializes}(t, \mathrm{2ndOT}) \quad \text{(T6)}$$

This leads to a basic pattern in the theory: every type that is not a basic type (e.g., a domain type) is an instance of one of the basic higher-order types ("1stOT", "2ndOT", and "3rdOT"), and, at the same time specializes the basic type at the immediately lower level (respectively, "Individual", "1stOT", "2ndOT"). For example, consider the enterprise domain, in which we may need a type to capture the concept of "Employee*"*. The type "Employee" classifies *individuals* (e.g. John or Mary), i.e., every *instance of* "Employee" is also *instance of* "Individual". Thus, by axiom A3, we have that "Employee" *is instance of* "1stOT" and, considering T4, "Employee" *specializes* "Individual". In fact, since "Employee" and "Individual" are different types, we can say that "Employee" *proper specializes* "Individual". This basic pattern is illustrated in Fig. 3. In order to preserve the intuition in the representation, we used the traditional UML notation to represent specializations (in this case to represent the fact that the proposition *properSpecializes(Employee*, *Individual)* holds). We have used the instance specification notation to represent an individual (John), while keeping the use of dashed arrows to show instantiation. The theory basic types are shaded to differentiate them from domain elements.

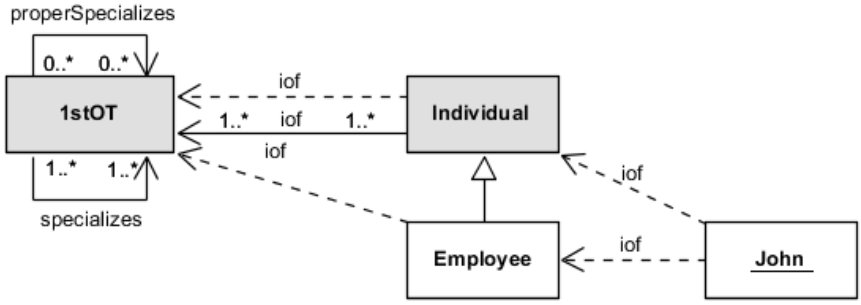

Fig. 3. Using the theory to model a domain.

Our theory supports also *specializations* and *instantiations* occurring between domain elements. For instance, supposing we need to classify the employees according to their highest academic degrees we can consider types such as "PhDEmployee" and "BachelorEmployee" to classify respectively employees having Ph.D. and bachelor degrees. These types are proper specializations of "Employee" since their instances are also instances of "Employee". Thus, by

the transitivity of specialization, they also specialize “Individual” and, considering theorem T4, they are instances of “1stOT”.

Futher, we may consider a *second-order* type called “EmployeeAcademicDegreeType” that have as instances the types that specialize “Employee” according to the academic degree (e.g “PhDEmployee” and “BachelorEmployee”). Again applying the basic pattern, “EmployeeAcademicDegreeType” *is instance of* “2ndOT” (since its instances are instances of “1stOT”) and *specializes* “1stOT” (see A4 and T5).

Fig. 4 augments Fig. 3 adding the discussed entities and relations. In order to increase the readability of the diagram, we use dashed rectangles to group elements that have a common link to other element and draw only one arrow between the border of the rectangle and the other element. For example, instead of representing two *iof* links between “EmployeeAcademicDegreeType” and its instances, we group its instances in a dashed rectangle and draw one *iof* link between such rectangle and “EmployeeAcademicDegreeType”. Moreover, we omitted the representation of some relations that are implied by the represented relations. For example, although we do not represent that “PhDEmployee” proper specializes “Individual” it can be inferred by the fact that it proper specializes “Employee” which, in turn, proper specializes “Individual”.

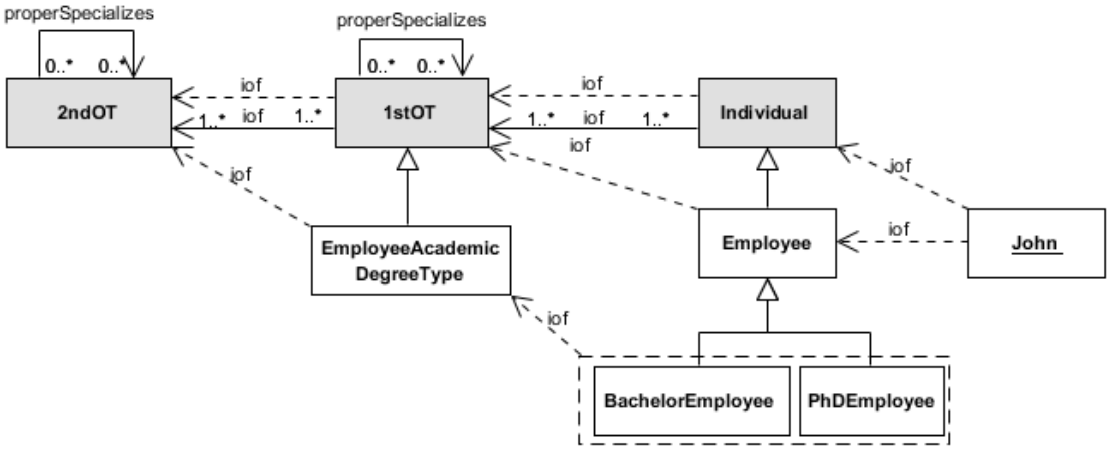

Fig. 4. Instantiations and specializations between domain elements.

Consider now an extension of the example in Fig. 4 in which we introduced a second *second-order type* called “EmployeeRoleType” beside “EmployeeAcademicDegreeType”. The instances of “EmployeeRoleType” are specializations of “Employee” according to the role they play (e.g. “Programmer” and “ResearchManager”). Consider further that, in order to reflect required qualifications in the domain, all instances of “EmployeeRoleType” must specialize instances of “EmployeeAcademicDegreeType”. In the example, we consider that “Programmer” *specializes* “BachelorEmployee” and “ResearchManager” *specializes* “PhDEmployee”. We can note that, a relation emerges between the “EmployeeRoleType” and “EmployeeAcademicDegreeType” higher-order types. We call this relation *subordination*.

A10 defines that *t1 is subordinate to t2* iff every *instance of t1* specializes an *instance of t2*. To avoid entities that have no instances to be trivially considered *subordinated* to other entities, A10 states that *subordination* does not apply to instances of "Individual".

$$\forall t1, t2\ \text{isSubordinateTo}\ (t1, t2) \leftrightarrow$$
$$(\neg \text{iof}(t1, \text{Individual}) \wedge (\forall t3\ \text{iof}(t3, t1) \rightarrow (\exists t4\ \text{iof}(t4, t2) \wedge \text{properSpecializes}(t3, t4)))) \qquad \text{(A10)}$$

Since *subordination* implies *specializations* between the instances of the involved types at one order lower, and *specializations* can only be established between types at the same order, *subordination* can only hold between *higher-order types* of equal order (see Fig. 5).

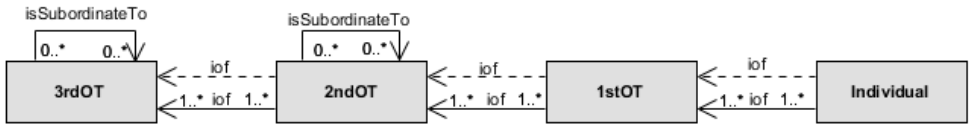

Fig. 5. Intra-level structural relations: subordination**.**

Fig. 6 illustrates the augmented example, showing that "EmployeeRoleType" *is subordinate to* "EmployeeAcademicDegreeType". Note that *subordination* between two higher-order types implies *specialization* between their instances but should be clearly distinguished from a specialization between the higher-order types (in the example, "EmployeeRoleType" does not specialize "EmployeeAcademicDegreeType").

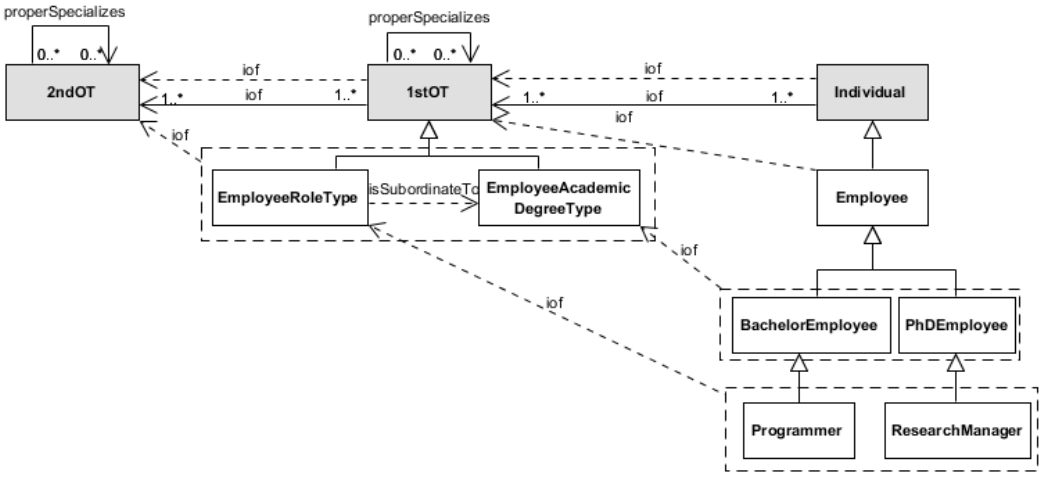

Fig. 6. An example of subordination relation.

Table 1 summarizes the characteristics of the defined intra-level structural relations.

Table 1. Intra-level structural relations characteristics

Name	Meaning	Domain and Range	Properties
Specialization *specializes(t1,t2)*	The principle of application of *t1* adds some classification criteria to the one of *t2* or both types have the same principle of application (t2=t1), i.e. every instance of *t1* is also an instance of *t2*.	Types of the same order (instances of 1stOT, 2ndOT or 3rdOT)	Reflexive, antisymmetric and transitive.
Proper Specialization *properSpecializes(t1,t2)*	The principle of application of *t1* adds some classification criteria to the one of *t2* i.e. every instance of *t1* is also an instance of *t2* and there at least one instance of t2 that is not instance of *t1*.		Irreflexive, asymmetric and transitive

Subordination *isSubordinateTo(t1,t2)*	The principle of application of each instance of *t1* adds some classification criteria to the principle of application of some instance of *t2* i.e. every instance of *t1* proper specializes some instance of *t2*.	Higher-order types of the same order (instances of 2ndOT or 3rdOT)	

4 Cross-level Structural Relations

This section defines the relations that occur between types of adjacent levels (the so-called *cross-level structural relations*). These relations support our analysis of the notions of *power type* in the literature, as well as their full incorporation in the theory.

The use of *power types* is one of the most common techniques for multi-level modelling. A seminal theory for the notion of *power type* was proposed by Cardelli [6]. According to [6], the same way specializations are intuitively analogous to subsets, *power types* can be intuitively understood as powersets. The powerset of a set A, is the set whose elements are **all** possible subsets of A including the empty set and A itself. Thus, "if A is a type, then Power(A) is the type whose elements are **all** the subtypes of A" (including A) [6]. Following Cardelli's definition, axiom A11 defines that iff a type *t1 is power type of* a type *t2* all *instances of t1* are *specializations of t2* and all possible *specializations of t2* are *instances of t1*. In this case, *t2* is said the base type of *t1*. Further, A11 guarantees that entities without instances (*individuals*) are not considered *power types of* other entities.

$$\forall t1, t2\ \text{isPowertypeOf}(t1, t2) \leftrightarrow (\neg \text{iof}(t1, \text{Individual}) \wedge (\forall t3\ \text{iof}(t3, t1) \leftrightarrow \text{specializes}(t3, t2))) \quad \text{(A11)}$$

Recall that "Individual" is an *instance of* "1stOT" (theorem T1) and that all the types that specialize "Individual" are also *instances of* "1stOT" (theorem T4). Thus, it follows from the definition of *powertype* (A11) that "1stOT" is *powertype of* "Individual" (theorem T7). Analogously, "2ndOT" is *powertype of* "1stOT" (theorem T8), and "3rdOT" is *powertype of* "2ndOT" (theorem T9).

$$\text{isPowertypeOf}(1\text{stOT}, \text{Individual}) \quad \text{(T7)}$$

$$\text{isPowertypeOf}(2\text{ndOT}, 1\text{stOT}) \quad \text{(T8)}$$

$$\text{isPowertypeOf}(2\text{ndOT}, 3\text{rdOT}) \quad \text{(T9)}$$

It is interesting to note that, to be a *power type*, a type must have a *principle of application* that defines that all its instances are specializations of the *base type* and, conversely, all specializations of the *base type* are instances of the *power type* (see A11). Thus, it is possible to conclude that each type has at most one *power type* (theorem T10) and that each *type* is *power type of*, at most, one other type (theorem T11). This suggests a concrete syntactic constraint for a multi-level model: only one power type can be linked to a base type through the *is power type of* relation.

$$\forall p, t\ \text{isPowertypeOf}(p, t) \rightarrow \nexists p'\ (p \neq p') \wedge \text{isPowertypeOf}(p', t) \quad \text{(T10)}$$

$$\forall p, t\ \text{isPowertypeOf}(p, t) \rightarrow \nexists t'\ (t \neq t') \wedge \text{isPowertypeOf}(p, t') \quad \text{(T11)}$$

Theorem T10 can be proved as follows: (i) supposing two higher order types, *p* and *p'*, are power type of *t*, according to A11, both *p* and *p'* should have as only instances all the specializations of *t*; (ii) thus, applying axiom A7, we conclude that *p* is equal to *p'* (*p=p'*). Analogously, theorem T11 can be proved as follows: (i) supposing *p is power type of t*, according to A11, *p* should have as only instances all the specializations of *t*; (ii) if we also consider a type *t'* such that *p is power type of t'* then *p* should have as only instances all the specializations of *t'*; thus, *t = t'*.

In his accounts for the notion of *power type* [6], Cardelli proved that if a type *t2 specializes* a type *t1* then the *power type of t2 specializes* the *power type of t1*. Since our definition for *isPowertypeOf* relation follows Cardelli's definition, we verified that this property is entailed by our theory. Theorem T12 formalizes this property. This theorem suggests a clear pattern for a multi-level model: the base type specialization hierarchy is isomorphic to the hierarchy of the power types. This may be used to check the syntax of power type hierarchies, and also to generate the power type hierarchy corresponding to the base type hierarchy.

$$\forall t1, t2, t3, t4\big(\text{specializes}(t2, t1) \wedge \text{isPowertypeOf}(t4, t2) \wedge \text{isPowertypeOf}(t3, t1)\big) \rightarrow \text{specializes}(t4, t3) \quad \text{(T12)}$$

T12 can be proved as follows: (i) considering that *t3 is powertype of t1* by axiom A11 we conclude that *t1* and all its specializations are *instance of t3*; (ii) considering the transitivity of specialization and that *t2 specializes t1*, we have that all specializations of *t2* also *specialize t1*, and thus, all specializations of *t2* are instance of *t3*; (iii) considering that *t4 is powertype of t2* by axiom A11 we conclude that all instances of *t4* are specializations of *t2*; (iv) thus, by (ii) and (iii) we conclude that all instances of *t4* are also instances of *t3*, i.e., *t4 specializes t3*.

Given the *power type* definition (A11), if *p1 is power type of t1* we conclude that *p1* is one order higher then *t1*, i.e., if *p1* is a *first-order* type (*iof(p1,1stOT)*) then *t1* is a second-order type (*iof(t1,2ndOT)*), if *p1* is a second-order type (*iof(p1,2ndOT)*) *t1* is a third-order type (*iof(t1,3rdOT)*), and so on. Furthermore, since instances of "Individual" are not types, they cannot participate in *isPowertypeOf* relations as *power type* nor as *base type*. Fig. 7 augments Fig. 1 by including the representation of *isPowertypeOf* relations.

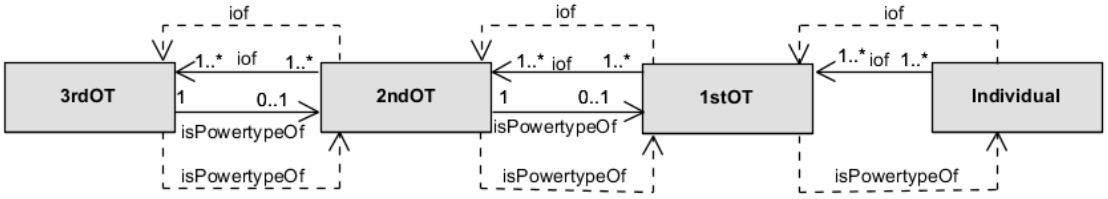

Fig. 7. Cross-level relations: *isPowertypeOf*.

Since the *power type* of a base type is a type whose principle of application defines that its instances classify instances of the base type, for each first-order type *f* it is always possible to define a second-order type *s* such that *s is power type of f* and for each second-order type *s* it is possible to define a third-order type *t* such that *t is powertype of s*. While the theory necessitates the existence of the power type of any type (except the power types of third-order types, which are outside the scope of the theory), the decision on whether to represent the *power type* of a particular type is a modelling decision. When the *power type* is not relevant for the domain being modelled it is often omitted from the model.

To illustrate the use the *is powertype of* relation, we augment the example of Fig. 6 in Fig. 8 introducing "EmployeeType", which *is powertype of* "Employee". Consequently, all types that *specialize* "Employee" are *instances of* "EmployeeType". Since the instances of "EmployeeType" are first-order types, "EmployeeType" is an *instance of* "2ndOT" and *specializes* "1stOT". Further, since all instances of "EmployeeRoleType" are also *instances of* "EmployeeType", it follows that "EmployeeRoleType" *specializes* "EmployeeType". Analogously, "EmployeeAcademicDegreeType" *specializes* "EmployeeType".

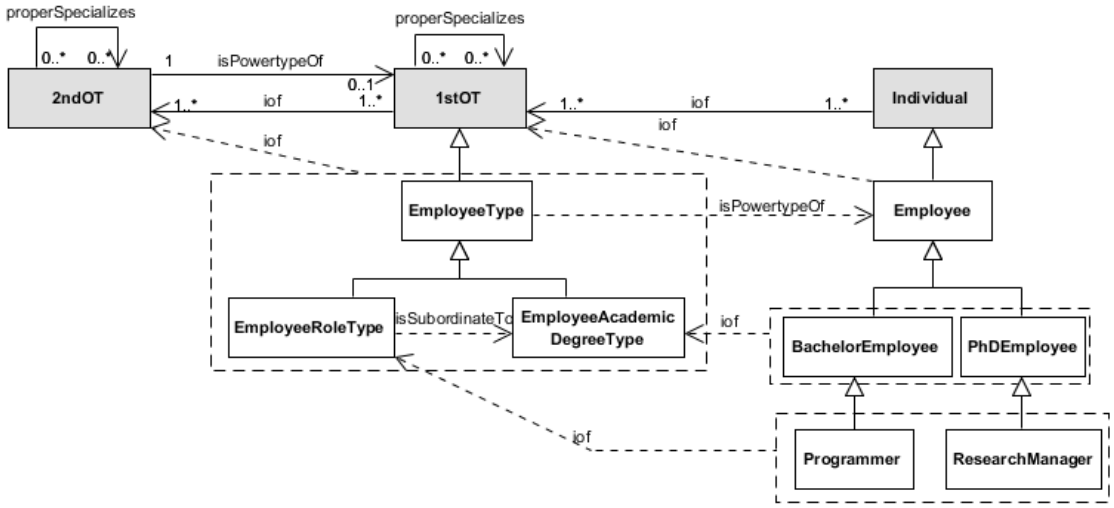

Fig. 8. An example of *isPowertypeOf* relation.

Although the definition of *power type* we adopted here is compliant with the one proposed by Cardelli [6], there are other definitions to this term in software engineering literature which have had great influence in practice, for example those definitions in [23,15].

In [23], Odell stated that a *power type* is a type whose instances are subtypes of another type. It is important to notice that Odell's definition is less strict than Cardelli's [6] definition. Cardelli follows the *power set* concept stating that **all** the specializations of the base type are instances of the *power type*. Odell's definition, in turn, does not comply with that restriction. Thus, as pointed out by [15], the relation defined by Odell is misnamed *power type* since, in fact, it denotes a *subset* of the *power set*.

Inspired on Odell's definition [23] we defined the *characterization* relation (Axiom A12): a type *t1 characterizes* a type *t2* iff all *instances of t1* are *properSpecializations* of *t2*. Further, A12 guarantees that *characterization relations* only apply to elements that are not *individuals*.

$$\forall t1, t2 \text{ characterizes } (t1, t2) \leftrightarrow (\neg \text{iof}(t1, \text{Individual}) \wedge (\forall t3 \text{ iof}(t3, t1) \rightarrow \text{properSpecializes}(t3, t2))) \qquad (A12)$$

The *characterization* relation occurs between a higher order type *t1* and a base type *t2* when the *principle of application* of *t1* defines that their instances *specialize t2* according to a specific *classification criteria*. Thus, the instances of *t1 specialize t2* but *t2* is not an *instance of t1* and there may be other types that *specializes t2* according to other *classification criteria* and, thus, are not *instances of t1*. *Characterization* relations only occur between types of adjacent levels (see Fig. 9).

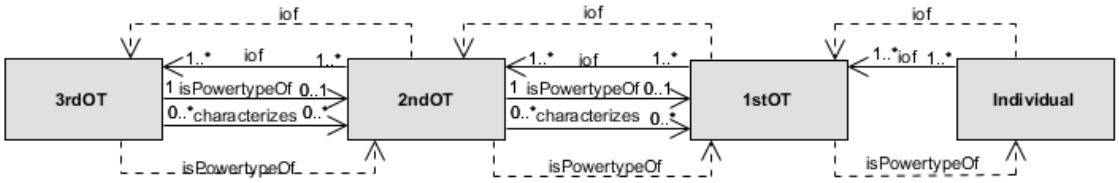

Fig. 9. Cross-Level relations: *characterizes*.

In our previous example, the instances of "EmployeeRoleType" *specialize* "Employee" according to roles the employee was hired to play (*the classification criteria)* whereas "EmployeeAcademicDegreeType" use the employees' academic degree as a criteria to classify employees. Thus, both "EmployeeRoleType" and "EmployeeAcademicDegreeType" *characterize* "Employee".

Considering the definitions of *power type* (axiom A11), *characterization* (axiom A12) and *proper specialization* (axiom A9) we conclude that if a type *t2* is *power type of* a type *t1* and a type *t3 characterizes* the same base type *t1* then all instances of *t3* are also *instances of* the power type *t2* and, thus, *t3 proper specializes t2*. This idea is formalized in theorem T13. This theorem can be used to check the completeness of models: a model would be incomplete if it omits the specialization between a type that characterizes a base type and this base type's power type.

$$\forall t1, t2, t3 \left(\text{isPowertypeOf}(t2, t1) \wedge \text{characterizes}(t3, t1)\right) \rightarrow \text{properSpecializes}(t3, t2) \qquad (T13)$$

Thus, both "EmployeeAcademicDegree" and "EmployeeRoleType" *characterize* "Employee" and *proper specialize* "EmployeeType".

In some cases, one needs more expressiveness in the description of the relation between higher-order type and characterized type. For instance, suppose that our company considers that each employee must play at least one role, i.e., in addition to the fact that "EmployeeRoleType" *characterizes* "Employee" the instances of "EmployeeRoleType" must completely classify the

instances of “Employee”. In order to accommodate this expressiveness we define a variation of *characterization* relation called *completeCharacterization* (see axiom A13). Thus, we are able to state that “EmployeeRoleType” *completely characterizes* “Employee”.

$$\forall t1, t2\ \text{completelyCharacterizes}(t1, t2) \leftrightarrow (\text{characterizes}(t1, t2) \wedge (\forall e\ \text{iof}(e, t2) \rightarrow \exists t3\ (\text{iof}(e, t3) \wedge \text{iof}(t3, t1)))) \quad \text{(A13)}$$

We also define a variation of *characterization* relation, called *disjointCharacterization*, to accommodate the cases in which each *instance of* the base type is *instance of* at most one instance of the characterizing type. Thus, according to A14, a type *t1 disjointlyCharacterizes t2* iff *t1 characterizes t2* and every instance of *t2* is *instance of*, at most, an *instance of t1*.

$$\forall t1, t2\ \text{disjointlyCharacterizes}\ (t1, t2) \leftrightarrow (\text{characterizes}(t1, t2) \wedge \forall e, t3, t4\ ((\text{iof}(t3, t1) \wedge \text{iof}(t4, t1) \wedge \text{iof}(e, t3) \wedge \text{iof}(e, t4)) \rightarrow t3 = t4))) \quad \text{(A14)}$$

In our example, we could consider that each employee falls under one classification according to his higher academic degree. Thus, “EmployeeAcademicDegreeType” simultaneously *disjointlyCharacterizes* and *completelyCharacterizes* “Employee”, i.e. each *instance of* “Employee” is *instance of* one and only one *instance of* “EmployeeExperienceType”. In this case we say that “EmployeeAcademicDegreeType” *partitions* “Employee” (see Fig. 10). Axiom A15 formally defines the *partition* relation.

$$\forall t1, t2\ \text{partitions}(t1, t2) \leftrightarrow (\text{completelyCharacterizes}(t1, t2) \wedge \text{disjointlyCharacterizes}(t1, t2)) \quad \text{(A15)}$$

The principle of application of a higher order type which *partitions* a base type defines that its instances must apply to instances of the base type and also define a classification criteria such that each instance of the base types is classified by one and only one instance of the higher order type.

Although the definition that Odell gave to the notion of *power type* is aligned with the relation we call *characterizes*, all examples of use provided in [23] exhibits relations that should be classified as *partitions* according to our theory. Henderson-Sellers [15], following those examples of use, provided a set theoretic formalization for the notion we call here *partition*.

Since all power type based relations (*powertype of*, *characterization*, *complete characterization*, *disjoint characterization* and *partition*) define that the instances of their domains are *specializations*/*proper specializations of* their ranges, both their domains and their ranges are types. Further, their domains must be a type in one order higher than their range. Thus, only higher-order types may play the role of domain of those power type based relations. Since *completely characterizes*, *disjointly characterizes* and *partition* relations all subsume *characterizes* relations, only the last one is represented in Fig. 10 for simplicity.

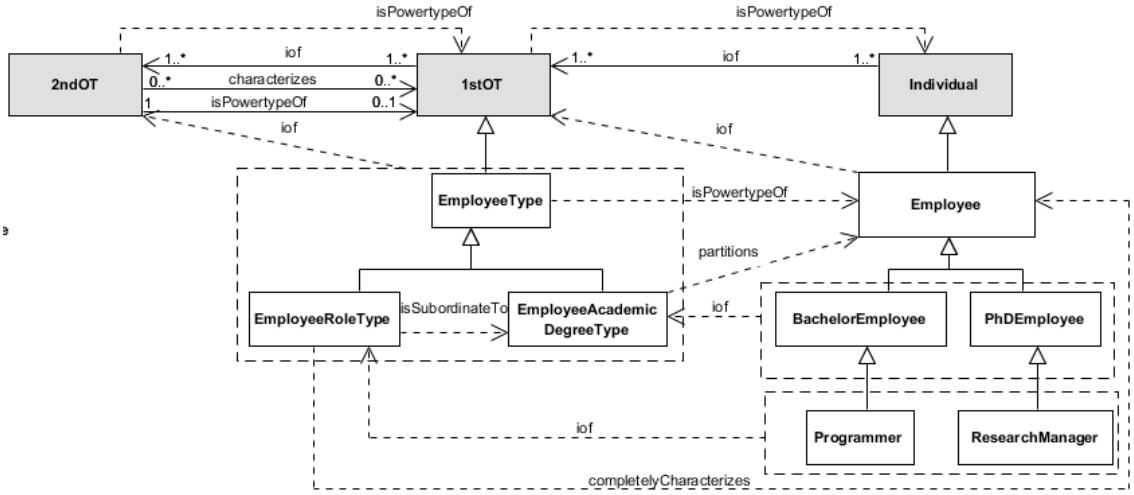

Fig. 10. An example of domain modelling applying *characterization* and *subordination* relations.

A consequence of the *partitions* definition is that, if two types *t1* and *t2* both partitions the same type *t3* then it is not possible for *t1* to specialize *t2*. This is captured in theorem T14. Again, this theorem suggests a clear syntactic constraint for a multi-level modelling language in the presence of more than one partition of the same base type.

$$\forall\, t1, t2, t3\ \big(\text{partitions}(t1, t3) \wedge \text{partitions}(t2, t3)\big) \rightarrow \neg\text{properSpecializes}(t1, t2) \quad \text{(T14)}$$

T14 can be proved as follows: (i) Using the definition of *partitions* (A15), we conclude that the instances of *t1* form a disjoint and complete partition of *t3*. (ii) Supposing *t1 proper specializes t2*, using the definition of *proper specialization* (A9) we conclude that all instances of *t1* must also be instances of *t2* and *t2* must have at least one additional instance that is not an instance of *p1*. (iii) Consider that t4 is the type that is instance of t2 and is not an instance of t1. Since t2 also partitions t3, then *t4* must specialize *t3*. (iv) However, the instances of *t2* that are also instances of *t1* already completely and disjoint classifies the instances of *t3*. Thus, *t4* does not have possible instances, and thus is not a valid type according to our theory. Therefore, there is no hypothesis in which *t1* partitions *t3*, *t2* partitions *t3* and *t1* specializes *t2*.

Table 2 summarizes some information about the cross-level relations. All these relations are irreflexive, asymmetric and intransitive.

Table 2. Cross-level structural relations characteristics

Name	Meaning	Domain and Range
Instantiation *iof(e,t)*	The principle of application of the *t* applies to *e*.	Elements of adjacent levels.
Powertype *isPowertypeOf(t1,t2)*	The principle of application of *t1* defines that its instances applies to instances of *t2* but does not define a *classification criteria*. Thus, the extension of *t1* is composed by all specializations of *t2*, including *t2* itself.	Types of adjacent levels (2ndOT→1stOT or 3rdOT→2ndOT)
Characterization *characterizes(t1,t2)*	The principle of application of *t1* defines that its instances applies to instances of *t2* according a specific *classification criteria*. Thus, the extension of *t1* is composed by the proper specializations of *t2* that follows the specified classification criteria.	
Complete Characterization *completelyCharacterizes(t1,t2)*	A variation of *characterization* in which the classification criteria defined by the principle of application of *t1* guarantees that each instance of *t2* is instance of at least one instance of *t1*.	
Disjoint Characterization *disjointlyCharacterizes(t1,t2)*	A variation of *characterization* in which the classification criteria defined by the principle of application of *t1* guarantees that each instance of *t2* is instance of at most one instance of *t1*.	
Partition *partitions(t1,t2)*	A variation of *characterization* in which the classification criteria defined by the principle of application of *t1* guarantees that each instance of *t2* is instance of exactly one instance of *t1*.	

5 Applying the theory to Taxonomical Structures

The previous section presented general implications of our theory for multi-level modelling. In this section we consider a representative application scenario in order to illustrate the theory expressiveness. We consider the biological taxonomy for living beings [18], which is one of the most mature examples of taxonomical hierarchies. The biological taxonomy for living beings classifies *living beings* according to *biological taxa* in seven or more ranks, e.g., *kingdom*, *phylum*, *class*, *order*, *genus*, *species*, and *breed*.

According to our theory every domain type is an instance of one of the basic higher-order types ("1stOT", "2ndOT", and "3rdOT"), and specializes the basic type at the immediately lower level (respectively, "Individual", "1stOT", "2ndOT"). Applying this pattern, we identify that (i) "LivingBeing" is an instance of "1stOT" and specializes "Individual" (since its instances are particular living beings), (ii) "BiologicalTaxon" and its specializations are instances of "2ndOT" and specializes "1stOT" (its instances are the first-order types which classify living beings, such as, e.g., the "Animalia" kingdom and the "Homo Sapiens" species[5]), and (iii) "BiologicalRank" specializes "2ndOT" and instantiates "3rdOT" (its instances are second-order

[5] Note that in biology there is a long and involved debate on the ontological status of taxa such as species [9]. One of the interpretations is that biological taxa (e.g., the "Homo Sapiens" species, the "Canis Lupus Familiaris" species) represents a group of animals rather than a kind or type of animal. We stay clear of this debate and represent species (and other taxa) as the type that is instantiated by all members of that group (and only by them) (e.g., "Human" and "Dog").

types which classify taxa, such as, e.g., the "Species" taxon). Fig. 11 shows a model for this domain using the basic pattern.

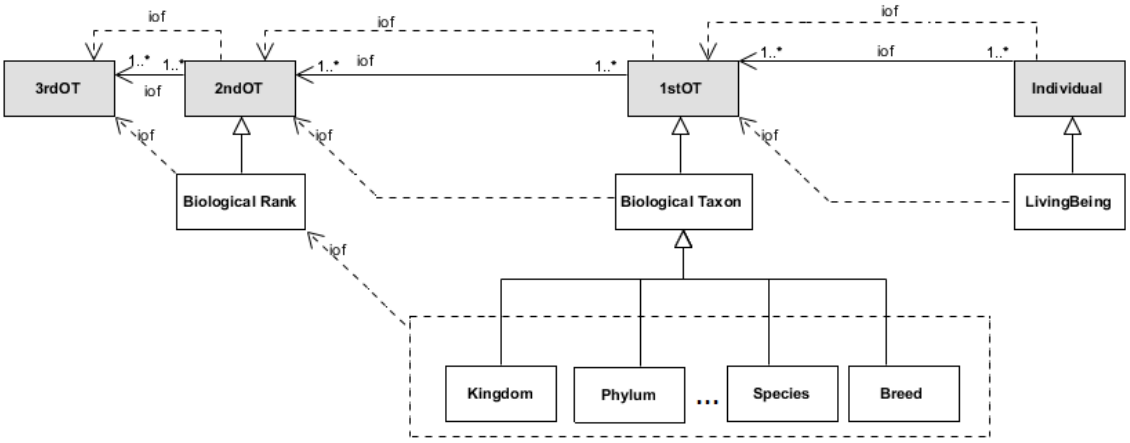

Fig. 11. Applying our theory basic pattern to the biological taxonomy for living beings.

Each "LivingBeing" is instance of one instance of each "Biological Rank", i.e., each living being is instance of one kingdom, one phylum, and so one. Therefore, we conclude that each one of the seven instances of "BiologicalRank" *partitions* "LivingBeing". Further, the instances of "Biological Rank" (*specializations* of "Biological Taxon") obey a subordination chain such that every instance of "Phylum" proper specializes one instance of "Kingdom", every instance of "Class" proper specializes one instance of "Phylum", and so on. Thus, according to our theory, each instance of "Biological Rank" *is subordinate to* another instance of "Biological Rank", forming a chain of subordination (except "Kingdom" which is the top of the chain). Since all instances of "Biological Rank" specialize "BiologicalTaxon" and each instance of "BiologicalTaxon" is instance of exactly one instance of "Biological Rank" (e.g., "Animal" is instance of "Kingdom", Collie is instance of Breed, etc.) according to our theory, "Biological Rank" *partitions* "BiologicalTaxon". Fig. 12 illustrates how the notions in the theory can be employed; one instance of each represented biological rank is shown.

This example of application shows the expressiveness of our theory. We have explored the entities and relations to fully describe the structural arrangement of the biological taxonomy for living beings.

The pattern to classify domain types as instantiations and specializations of the theory basic types permitted us to identify the level of each involved concept. Using the notion of partition relation we were able to (i) express how the instances of biological rank apply to living beings and (ii) to understand the relation between biological rank and biological taxon. The notion of *subordination* relation was central for understanding how the instances of biological rank are related to each other.

Finally, it allowed us to notice that the shape of tree that the biological taxonomy for living beings exhibits is explained by the combination of two characteristics, namely, (i) the *partitions*

relations that all instances of “BiologicalRank” have with “LivingBeing”, and (ii) the chain of *subordination* that the instances of “BiologicalRank” forms.

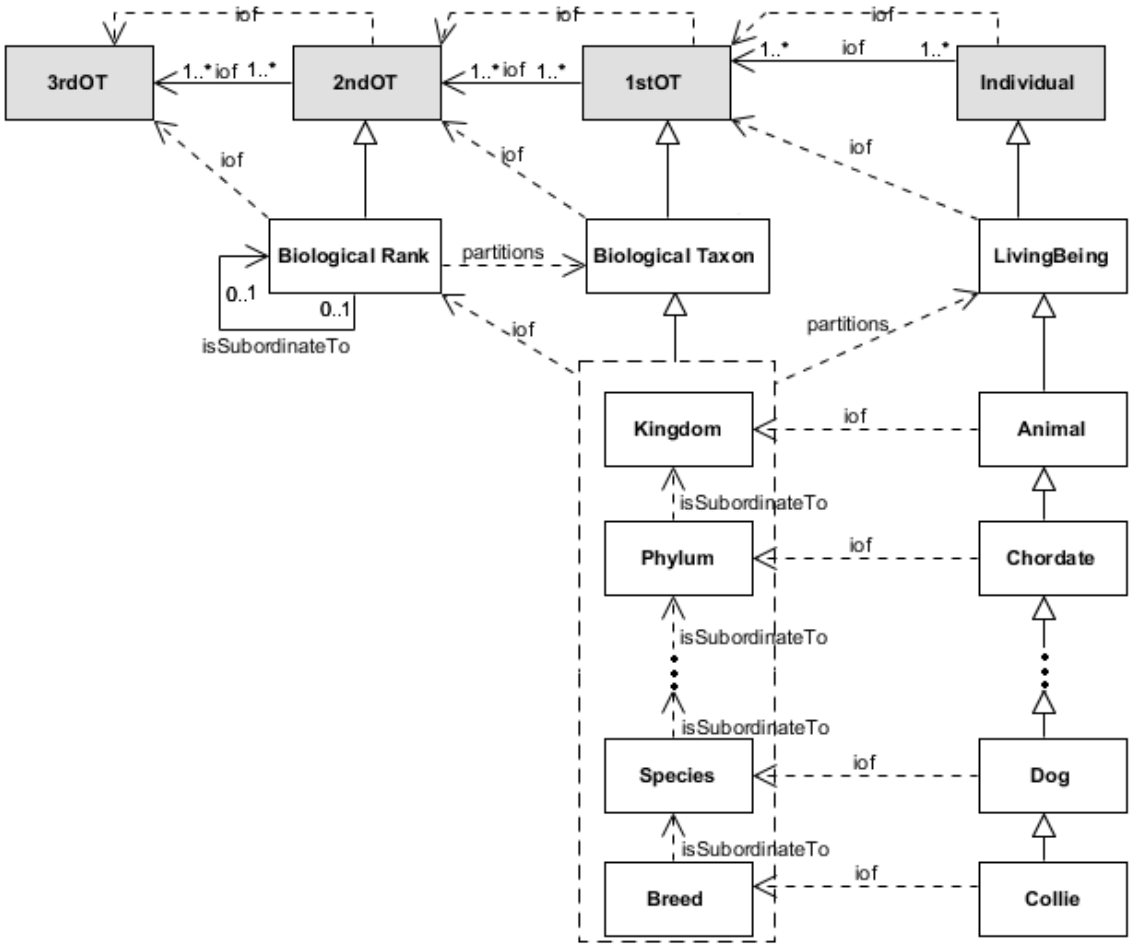

Fig. 12. Using our theory to describe the structural relations that exist in biological taxonomy (relations between the notions of biological rank, biological taxon and living being).

6 Related Work

6.1 Power type-based approaches

Two early attempts to address multi-level modelling, namely *power types* [6, 23] and *materialization* [27], raised from the identification of patterns to represent the relationship between a class of categories and a class of more concrete entities. The notion of *power types* was adopted in the object-oriented model community (largely influenced by [23]) and *materialization* has been developed in the database community. Despite the different origins, *power type* and *materialization* are based on similar conceptualizations [3] and addressing the same concerns [11]. Both approaches establish a relationship between two types such that the instances of one are specializations (subtypes) of another.

Odell [23] defined the concept of power type informally using regular associations between a class representing the power type and a base class. This differs from our approach because

cross-layer relations between types (*is power type of*, *characterizes* and *partitions*) have specialized semantics. This allows us to prescribe rules for the domain models that use these relations following the axioms in the theory.

Similarly to Odell [23], Gonzalez-Perez and Henderson-Sellers [11] use an association labelled "partitions" between a power type and a base type (called a "partitioned type" in their terminology). The authors illustrate their technique with a diagram in which "partitions" is modelled as a one-to-many association between "Task" and "TaskKind", meaning that every instance of the partitioned type ("Task") is linked to exactly one instance of the powertype ("TaskKind"). In the sequel, they discuss that the "*partitions* association possesses instantiation semantics", and that, because of this, "Task" is a special instance of "TaskKind" (the most generic kind of task). However, if "Task" itself is an instance of "TaskKind", then the "partitions" association cannot be a one-to-many association between "Task" and "TaskKind". This is because all instances of subtypes of "Task" are also instances of "Task", and thus instances of at least *two* "TaskKinds" (one which is "Task" itself). The source of the difficulty seems to lie in that their "partitions" association is semantically overloaded, conflating two underlying notions: (i) the fact that "TaskKind" *partitions* "Task", and (ii) the implied consequence that instances of "Task" are instances of instances of "TaskKind" (which in our theory is reflected in the *instance of* relation between "Task" as specialization of "Individual" and "TaskKind" as a specialization of "First-Order Type"). The modeller is free to determine whether "Task" itself is an instance of "TaskKind" (in which case he/she would replace (i) with the fact that "TaskKind" *is a powertype of* "Task"). Note that the elements of our theory help us to identify the semantic overload, provide an explanation for the conceptual issue in this power type based approach, and offer alternatives to express the modeller's intended conceptualization.

The UML 2.4.1 specification [25] attempts to cover the needs of multi-level modelling by including a *powertype 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 powertype. Because of its dependence on the generalization set construct, the pattern can only be applied when specializations of the base type are explicitly modelled (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 "employee type" as a powertype of "employee", without forcing the modeller to define specific instances for "employee type". While our theory necessitates the existence of entities for any type, and hence necessitates the existence of instances for "employee type", it does not require these instances to be modelled explicitly, which is the case of the UML because of its choice to base the power type pattern in a structure that uses generalization sets.

Further, while the *completelyCharacterizes* relation is similar to the *isCovering* attribute of *GeneralizationSets* of the UML metamodel, there is an important distinction. The attribute *isCovering* refers to whether all instances of the general classifier are instances of at least one of the specific classifiers that are explicitly modelled in the *GeneralizationSet*. In contrast, *completelyCharacterizes* is a semantic notion that is independent of what is represented explicitly in a model; when a higher-order type is related to a base type through this relation, all instances of the base type will be instances of at least one of the types that properly specialize the base type. This makes *isCovering* restricted to those domains that can afford an explicit enumeration of the instances of the higher-order type.

A semantic mapping of UML's *isCovering* attribute in terms of our theory is simple when *isCovering* is true, in which case the higher-order type *p completely characterizes* the base type *t*. However, when *isCovering* is false, the semantic mapping is more involved, and the model can have two alternative interpretations: (i) *p characterizes t* and there are instances of *p* not represented in the model (it is not possible to determine whether *p completely characterizes t*); or (ii) *p characterizes t* and all instances of *p* are represented, but some instances of *t* do not instantiate any of the instances of *p* (thus, we conclude that *p* does not *completely characterize t)*. The lack of expressiveness of UML to distinguish these interpretations seems to stem from the fact that UML conflates what we mean to capture with the *characterizes* relation with whether the *model enumerates all instances of a characterizing higher-order type* (i.e., the "power type" in UML's terminology).[6]

The notion of *power type* introduced by Odell [23] in the object oriented community differs from the concept coined earlier by Cardelli [6] since the latter is derived directly from the mathematical notion of *power set* while the former may be used more loosely as we discussed in section 4. The theory presented here is able to account for both definitions formally, revealing their differences. It covers the expressiveness of both approaches through formally-defined structural cross-level relations (*is power type of, characterizes* and *partitions*). Further, it allows us to show that a higher-order type that is related to a base type through the *characterizes* relation is necessarily a specialization of the power type of that base type. Thus, the power type of a base type is the most abstract higher-order type related to a base type.

6.2 Clabject-based approaches and deep instantiation

In [1], Atkinson and Kühne argue that a multi-level modelling framework should adhere to two fundamental principles: *strict metamodelling* and support for the *clabject* notion.

Strict metamodelling [5] assumes that each element of a level must be an instance of an element of the level above. Although our theory is not focused on metamodelling, we follow

[6] For the sake of simplicity we have assume here that the classes are not abstract. The semantic mapping becomes even more involved in the presence of abstract classes.

this principle, and every entity in our domain of enquiry is instance of exactly one of the basic types and every entity can only be instance of entities at one order higher (all entities that have no instances are instances of "Individual"; "Individual" and all its specializations are instances of "First-Order Type", and so on.) They also discuss that "some kind of 'trick' is needed at the top level". The 'trick' we used in our theory is that the highest order foundational type is not instance of anything, since entities with higher order are not considered (see axioms A5 and A6). Alternatively, an infinite number of types may be considered, and a 'trick' is not needed.

Besides adhering to the principle of *strict metamodelling*, the notion of *clabject* [1] is also valuable for our theory. This notion is founded on the observation that every instantiable entity has both a type (or class) facet and an instance (or object) facet. The basic types of our theory, except the higher order one, may be considered *clabjects*. For example, "Individual" is instance of "First-Order Type" (its instance facet) and a type for all entities that are not types (its type facet).

Atkinson and Kühne have also proposed a *deep instantiation based* approach [4, 3] 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. It 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. Therefore, instances of *mobile phone model* would be *clabjects* in which *IMEI* attribute would have potency of 1. Instances of instances of *mobile phone model* have a value assigned to *IMEI*, since its potency would reach zero.

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 modelling without the need of introducing types to the models only "because of the idiosyncrasies of a particular solution to deep characterization" [3]. They argue that *power type* based solutions force the modellers to add unneeded types to the model. For instance, considering the cited example of *mobile phone model*, using *power types* the modeller would be forced to represent the concept of *mobile phone*. Using deep instantiation, the modeller could define the *mobile phone* properties (e.g. *IMEI)* as properties of *mobile phone type* having potency of 2, being free to not represent the concept of *mobile phone*.

While the deep instantiation approach can reduce the number of entities represented in a model, this strategy should be used with parsimony. This is because whenever properties with

potency higher than one are used to omit a base type, the clabjects which instantiate the higher-order type "inherit" the properties with potency higher than one from the higher-order type. In this case, the instantiation relation is overloaded with an implicit specialization relation, and semantic clarity is traded for reduction of model size. Further, as discussed in [14], conceptual models should always include types that define the principle of identity of individuals (in the example this type is *mobile phone*). If types are omitted (and incorporated into higher-order types by using the notion of potency), the source of principle of identity becomes hidden. Another important consequence of omitting a base type is that we become unable to express whether the instances of a higher-order type are disjoint types (i.e., we are unable to distinguish which form of *characterization* would apply.)

Another multi-level modelling approach proposes the concepts of *m-objects* and *m-relationships* [21]. The focus of this approach is also on reducing "unnecessary complexity", improve readability and simplify maintenance and extension. *M-objects* encapsulate different levels of abstraction that relate to a single domain concept. Analogously, *m-relationships* describe "relationships between *m-objects* at multiple level of abstraction". An *m-object* can *concretize* another *m-object*. The *concretize* relationship comprises classification, generalization and aggregation relationships between the levels of an *m-object* [21]. We observe that this is a semantic overload between three relationships of quite different ontological nature, which could affect the understandability and usability of the approach.

In [2] the authors point out the need of considering two different kinds of instantiation: the linguistic instantiation and the ontological instantiation. Whereas linguistic instantiation is used to define the relations between domain entities and linguistic constructs, ontological instantiations relate domain entities to other domain entities. For example, considering the UML class diagram having a *class* called *Collie* and an *object* called *Lassie*, we can identify two linguistic instantiations, namely, *Collie* is an (linguistic) instance of *class* and *Lassie* is an (linguistic) instance of *object*. We may also consider an ontological instantiation since *Lassie* is an (ontological) instance of *Collie* [2]. The distinction is very important to determine the scope of our theory: we are concerned solely with ontological instantiation as it is applied across multiple levels. For example, the instantiation relation that holds between *individual* and *first-order type* as well as the one that holds between *computer* and *first-order type* are both ontological since both are concerned with the nature of the involved concepts and none of them is related to linguistic issues. Aspects referring to the relation between ontological and linguistic issues are out of the scope of this work.

7 Final Considerations

In this paper we have presented a well-founded theory for conceptual multi-level modelling. The theory is formally defined using first-order logic and its consistency is verified using a lightweight formal method. Both the basic types and the structural relations defined in the theory are founded on the basic notion of (ontological) instantiation, which is applied regularly across levels, following the principle of strict (meta-)modelling. We have shown how the elements of the theory can be used as foundations for a domain theory: domain types instantiate and specialize the basic types of the theory.

To verify the consistency of our theory we have used Alloy [16]. The axioms of our theory were represented as facts and the theorems were defined as assertions in an Alloy module. It allowed us to verify the satisfiability of our theory, to conduct some model simulations and to verify the theorems whose informal proofs have been discussed in the paper.[7]

Using the structural cross-level relations defined in the theory (*is powertype of*, *characterizes*, *partitions*), we are able to account for the different notions of power type in the literature, as well as to contrast and relate them. Since these relations are ultimately explained in terms of instantiation between entities of adjacent levels, the consequence of our account of power types is that we formally harmonize power type and clabject-based approaches.

With respect to intra-level relations, we define the "ordinary" specialization relation and a *subordination* relation between higher-order types of the same order. Subordination allows for the creation of expressive multi-level models; subordination between higher-order types implies specialization between instances of the types related by subordination. An example of the usefulness of the subordination relation is shown in the biological taxonomy domain, in which taxonomic ranks (instances of "Second-Order Type") are related by subordination in a sequence (with lower ranks subordinated to higher ranks). This ensures the taxonomy at the first-order level has an adequate structure (a taxonomic tree).

We have opted to supress two direct extensions of the theory in order to facilitate its presentation in this paper: (i) the support for non-rigid types, and (ii) the generalization of the notion of order to support an infinite number of classification levels. With respect to (i) this extension can be performed by including a third parameter in the *iof* predicate representing a possible world or state of the world. This would allow instantiation to be contingent, thereby enabling dynamic classification, which is an important feature for conceptual modelling [14]. With respect to (ii), axioms A3 and A4 would give way to an inductive definition for a basic type T_{i+1} based on the definition of the basic type at an immediately lower order T_i. The "disjointness" and "completeness" axioms (A5 and A6) would be modified accordingly.

[7] The full specification of the theory in Alloy can be found in https://github.com/jpalmeida/mlt-ontology

It is important to stress that it is not our intention in this paper to propose a multi-level conceptual modelling language. Instead, we focus on the concepts that would constitute an adequate semantic domain for such a language. The theory we propose can be considered a reference top-level ontology for types, with the main purpose of clarifying key concepts and relations for multi-level conceptualizations.

As discussed in [14], a reference ontology can be used to inform the revision and redesign of a modelling language, first through the identification of semantic overload, construct deficit, construct excess and construct redundancy, but also through the definition of modelling patterns and semantically-motivated syntactic constraints [7]. This has been fruitful in the past in the revision of the UML, resulting in the OntoUML profile for conceptual modelling [14]. Thus, a natural application for this work is to inform the (re-)design of a well-founded multi-level conceptual modelling language. Some earlier results to that extent are presented in section 4, showing: (i) how theorems of the theory reveal useful syntactic constraints for multi-level domain models; and (ii) how patterns of domain entities that are admissible by the theory can be reflected in modelling patterns. We have also been able to spot a deficiency in the UML given its reliance on the construct of generalization set to represent the power type pattern. Further, we have been able to identify cases of semantic overload in the power-type based technique presented in [11], in the deep instantiation technique [4] and in the m-objects approach [21]. Recently, Recker et al. [28] reported results from a study with 528 modellers demonstrating that "users of conceptual modelling grammars perceive ontological deficiencies to exist and that these deficiency perceptions are negatively associated with usefulness and ease of use of these grammars". This highlights the potential practical implications of our theory.

We are currently working on an extension of the Unified Foundation Ontology (UFO) [14] to fully incorporate the theory presented in this paper. The current version of UFO only counts with an informal notion of higher-order universal, with no associated formalization. The theory would serve as the top-most layer of UFO, and the typology of universals of UFO would be incorporated as specializations of "First-Order Type", including RigidUniversal, Anti-RigidUniversal, Category, Kind, Role, Phase, etc. Further, "Individual" would be specialized into Endurant, Moment, Event, Action, etc., leveraging important conceptual distinctions of UFO. The revision of UFO to incorporate this theory will give us a sound basis to improve the formalization of ontologies based on UFO (e.g., the core ontology for services called UFO-S [20] and the organizational ontology called O3 [26]) since their conceptualizations span multiple levels of classifications.

References

1. Atkinson, C., Kühne, T.: Meta-level Independent Modeling. International Workshop "Model Engineering" (in conjunction with ECOOP'2000), Cannes, France (2000)
2. Atkinson, C., Kühne, T.: Model-driven development: a metamodeling foundation. IEEE Software. 20(5), 36–41 (2003)
3. Atkinson, C., Kühne, T.: Reducing accidental complexity in domain models. Software & Systems. Modelling, 7(3), pp 345-359. Springer-Verlag (2008)
4. Atkinson, C., Kühne, T.: The Essence of Multilevel Modeling. In: Proc. Of the 4th International Conference on the Unified Modeling Language. Toronto, Canada (2001)
5. Atkinson, C.: Metamodelling for Distributed Object Environments. First International Enterprise Distributed Object Computing Workshop (EDOC'97). Brisbane, Australia (1997)
6. 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)
7. Carvalho, V. A., Almeida, J.P.A., Guizzardi, G.: Using Reference Domain Ontologies to Define the Real-World Semantics of Domain-Specific Languages. In: Proc. 26th International CAiSE Conference (CAiSE 2014), Heidelberg: Springer, 2014. pp. 488-502 (2014)
8. Coquand, T.: Type Theory, The Stanford Encyclopedia of Philosophy (Fall 2014 Edition), Edward N. Zalta (ed.), URL = http://plato.stanford.edu/archives/fall2014/entries/type-theory/ (2014)
9. Ereshefsky, M., Species, The Stanford Encyclopedia of Philosophy (Spring 2010 Edition), Edward N. Zalta (ed.), URL = http://plato.stanford.edu/archives/spr2010/entries/species/ (2010)
10. 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)
11. Gonzalez-Perez, C., Henderson-Sellers, B.: A powertype-based metamodelling framework. Software & Systems. Modelling, 5(1), 72-90. Springer-Verlag (2006)
12. Guarino, N., Welty, C.: Evaluating Ontological Decisions with OntoClean. In Communications of the ACM, 45(2), pp.61-65. (2002)
13. Guarino, N.: The Ontological Level. In: R. Casati, B. Smith and G. White (eds.), Philosophy and the Cognitive Science, pp. 443-456. Holder-Pivhler-Tempsky, Vienna (1994)
14. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models. University of Twente, Enschede, The Netherlands (2005)
15. Henderson-Sellers, B.: On the Mathematics of Modeling, Metamodelling, Ontologies and Modelling Languages. Springer (2012)
16. Jackson, D.: Software Abstractions: Logic, Language and Analysis. The MIT Press, (2006)
17. Kühne, T.: Contrasting Classification with Generalisation. In: Proc. of the 6th Asia-Pacific Conference on Conceptual Modeling. Wellington, New Zealand (2009)
18. Mayr, E., The Growth of Biological Thought: Diversity, Evolution, and Inheritancehe Belknap Press, 1982.
19. Mylopoulos, J.: Conceptual Modeling and Telos. In: Loucopoulos, P., Zicari, R. (Eds.), Conceptual modeling, databases and CASE, pp. 49-68, Wiley(1992)
20. Nardi, J.C., Falbo, R., Almeida, J.P.A., Guizzardi, G., Ferreira Pires, L., van Sinderen, M., Guarino, N.: Towards a Commitment-Based Reference Ontology for Services. In: Proc. 17th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2013), IEEE Computer Society Press, pp. 175-10 (2013).
21. Neumayr, B., Grün, K., Schrefl, M.: Multi-level domain modeling with m-objects and m-relationships. In: Proc. of the 6th Asia-Pacific Conference on Conceptual Modeling. Wellington, New Zealand (2009)
22. Neumayr, B., Schrefl, M., Thalhiem, B.: Modeling Techniques for Multi-Level Abstraction. In: Kaschek, R., Delcambre, L. (eds.). LNCS, vol. 6520, pp 68-92. Springer, Heidelberg(2011)
23. Odell, J.: Power types. In: Journal of Object-Oriented Programing, 7(2), pp. 8-12.(1994)
24. Olivé, A.: Conceptual Modeling of Information Systems. Springer (2007)
25. OMG : UML Superstructure Specification – Version 2.4.1 (2011)
26. Pereira, D., Almeida, J.P.A.: Representing Organizational Structures in an Enterprise Architecture Language. In: Proceedings of the 6th Workshop on Formal Ontologies meet Industry (FOMI 2014), Rio de Janeiro (2014).
27. Pirotte, A., Zimanyi, E., Massart, D., Yakusheva, T.: Materialization: a powerful and ubiquitous abstraction pattern. In: Bocca, J.,Jarke, M., Zaniolo, C. (eds.) Procs. 20th Int. Conf. Very Large DataBases (VLDB '94) pp. 630–641 (1994)
28. Recker, J., Rosemann, M., Green, P., Indulska, M.: Do Ontological Deficiencies in Modeling Grammars Matter?. In MIS Quarterly, 35 (1): pp. 1–9. (2011)