Using a Well-Founded Multi-Level Theory to Support the Analysis and Representation of the Powertype Pattern in Conceptual Modeling

Victorio Albani Carvalho[1,2], João Paulo A. Almeida[1], and Giancarlo Guizzardi[1]

[1]Ontology & Conceptual Modeling Research Group (NEMO)
Federal University of Espírito Santo (UFES), Vitória, ES, Brazil
[2]Federal Institute of Espírito Santo (IFES), Colatina, ES, Brazil
victorio@ifes.edu.br; jpalmeida@ieee.org; gguizzardi@inf.ufes.br

Abstract. Multi-level conceptual modeling addresses the representation of subject domains dealing with multiple classification levels. In such domains, the occurrence of situations in which instances of a type are specializations of another type is recurrent. This recurrent phenomenon is known in the conceptual modeling community as the powertype pattern. The relevance of the powertype pattern has led to its adoption in many important modeling initiatives, including the UML. To address the challenge of multi-level modeling, we have proposed an axiomatic well-founded theory called MLT. In this paper, we demonstrate how MLT can be used as a reference theory for capturing a number of nuances related to the modeling of the powertype pattern in conceptual modeling. Moreover, we show how this theory can be used to analyze, expose limitations and redesign the UML support for modeling this pattern.

Keywords: conceptual modeling, multi-level modeling, powertype, UML.

1 Introduction

Three fundamental quality attributes that must be reinforced in all conceptual modeling languages are *expressivity, clarity* and *parsimony* [1]. The first refers to the ability of the language to capture all relevant aspects of the phenomena in reality it purports to represent; the second to how easy it is for the language users to unambiguously recognize which aspects of the underlying phenomena are represented; the third to how economic a language is in not forcing the modeler to represent more than it is necessary for a problem at hand. There is now a long tradition in conceptual modeling of using *Reference Theories* to evaluate and (re) design conceptual modeling languages according to these quality attributes [2]. Examples of fundamental conceptual modeling constructs that have been analyzed and re-designed following this strategy include types and taxonomic structures, part-whole relations, intrinsic and relational properties, roles, etc. [3]. In this paper, we address a fundamental conceptual modeling notion, which is recurrent in a multitude of application domains, namely, the notion of *multi-level classification*.

In several subject domains, 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, to describe the conceptualization underlying the software development domain, one needs to represent entities of different (but nonetheless related) classification levels, such as tasks (i.e., specific events occurring in time and space), types of tasks, and types of types of tasks. Other examples of multiple classification levels come from domains such as biological taxonomy [4] and product types [5]. 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 modeling [5, 6]. Techniques for multi-level conceptual modeling must provide modeling concepts to deal with types in various classification levels and the relations that may occur between those types. The interest in multi-level modeling has led to a number of research initiatives in this subject (e.g. [5, 6, 7, 8, 9]).

In this paper, we address an early and important approach for multi-level modeling named the *powertype pattern* [8, 9]. This pattern is manifested when we have a case in which the instances of a type (the so-called "powertype") are specializations of a lower-level type (the so-called "base type"). Take, for instance, the case in which we want to express that the instances of the powertype *Bird Species* are types that specialize the base type *Bird*, including the particular subtypes of bird *Golden Eagle* and *Emperor Penguin*. The powertype pattern is extensively used in many important modeling initiatives. An example is the ISO/IEC 24744 standard [10]. Moreover, this pattern can regularly be found in many catalogues of modeling best practices, in which it appears as an ingredient of other patterns (see, for instance, [11]). Finally, the relevance of this pattern has led to its adoption in the current version of the Unified Modeling Language (UML) [12], which allows modelers to specify a powertype in the context of a "generalization set".

Despite its intuitive characterization, there are many important and subtle aspects of the relation between the "powertype" and "base type" in the powertype pattern that are often neglected. For instance, when stating that the instances of Bird Species are subtypes of Bird, do we mean that all subtypes of Bird are instances of Bird Species? Do we mean that only subtypes of Bird are instances of Bird Species? Both? Moreover, if the powertype can have as instances the subtypes of the base type then it follows that the instances of the base type can be *instances of instances* of the powertype. But does that mean, for example, that all instances of Bird must be instances of at least one instance of Bird Species? Does that mean that the instances of Bird cannot be instances of more than one instance of Bird Species? Both?

In order to explore the semantic issues involving the powertype pattern, we employ here a multi-level modeling theory called MLT, which we have proposed originally in [13]. MLT is a formal axiomatic theory founded on the notion of (ontological) instantiation and is able to: (i) clarify and position conflicting definitions of powertype in the literature; and, (ii) enrich the expressivity of multi-level modeling primitives, by defining new structural relations for variants of the powertype pattern. So far, we have used MLT in order to provide conceptual foundations for dealing with types at different levels of classification in core [14] and foundational ontologies [15]. Here we apply MLT to analyze the UML support for modeling the powertype pattern.

UML is a *de facto* standard for conceptual modeling and information systems engineering. Moreover, it is the basis for ontology-driven conceptual modeling languages such as OntoUML [3], which in the past years have gained increasing adoption in the conceptual modeling and ontology engineering communities [16]. For this reason, we believe that providing precise and unambiguous semantics and advancing the UML support for modeling powertypes amounts to an important contribution for conceptual modeling, in general, and for ontology-driven conceptual modeling and ontology engineering, in particular.

By using MLT as a well-founded *reference theory*, we analyze and expose a number of limitations in the existing UML support for modeling the powertype pattern. In particular, we demonstrate that this support: (i) lacks *expressivity*, for example, for representing different definitions of powertype that exist in the literature [8, 9], each of which has relevant applications; (ii) that it lacks *clarity*, for example, because it confounds constraints that apply to powertype instantiation with those that apply to corresponding generalization sets; (iii) that it lacks *parsimony*, for example, because it forces the modeler to explicitly represent at least one instance of each powertype. By employing the results of this analysis, we propose a UML profile for addressing the exposed limitations. We use the distinctions put forth by MLT to devise this profile and we use the formal rules inherent to MLT to guide the development of the profile's syntactic constraints.

The remainder of this paper is structured as follows: section 2 reviews briefly the MLT multi-level theory; section 3 discusses UML's current support for powertypes, revealing its limitations in light of MLT; section 4 presents our proposal to extend a fragment of UML reflecting the rules of MLT; section 5 discusses related work and section 6 presents concluding remarks.

2 MLT: A Theory for Multi-Level Modeling

MLT is formally defined using first-order logic, quantifying over all possible entities (individuals and types). According to MLT, types are predicative entities that can possibly be applied to a multitude of entities (e.g. Person, Car, Student). If a type *t* applies to an entity *e* then it is said that *e is an instance of t*. In contrast, particular entities, that have no instances, are considered *individuals* (e.g. John, Lassie, my car).

The *instance of* relation is represented in this formal theory by a binary predicate *iof(e,t)* that holds if an entity *e is instance of* an entity *t* (denoting a type). MLT admits types having individuals as instances as well as types that have other types as instances. In order to accommodate these varieties of types, the notion of *type order* is used. Types having individuals as instances are called *first-order types*, types whose instances are first-order types are *second-order types* and so on.

The logic constant "Individual" is used to define the conditions for entities to be considered individuals: *an entity is an instance of "Individual" iff it does not have any possible instance* (Axiom A1 in Table 1). The constant "First-Order Type" (or shortly "1stOT") *characterizes the type that applies to all entities whose instances are instances of "Individual"* (A2 in Table 1). Analogously, each *entity whose possible*

extension contains exclusively instances of "1stOT" is an instance of "Second-Order Type" (or shortly "2ndOT") (A3 in Table 1).

It follows from axioms A1, A2 and A3 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) (A4 in Table 1). 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 [13].

Some structural relations to support conceptual modeling are defined in MLT. According to MLT, a type *t specializes* another type *t'* iff *all instances of t are also instances of t'* (see definition D1 in Table 1). Since the reflexivity of the *specialization* relation may be undesired in some contexts, we define in MLT the *proper specialization* relation as follows*: t proper specializes t' iff t specializes t' and t is different from t'* (see D2 in Table 1). The definitions presented thus far guarantee that both *specializations* and *proper specializations* may only hold between types of the same order.

From the axioms and definitions presented so far one can conclude that every type that is not one of MLT's basic types (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") [13]. For example, consider a type "Person" that applies to all human beings. Since "Person" applies to individuals, it is an instance of "1stOT" and proper specializes "Individual". Further, consider a type named "Person Age Phase" whose instances are specializations of "Person" (thus, instances of "1stOT") that classify persons according to their age (e.g. "Child" and "Adult"). Thus, "Person Age Phase" is instance of "2ndOT" and proper specializes "1stOT".

MLT also defines relations that occur between types of adjacent orders, the so-called *cross-level structural relations*. These relations support an analysis of the notions of powertype in the literature.

One prominent notion of powertype was proposed by Cardelli [9]. According to Cardelli, the same way specializations are intuitively analogous to subsets, *powertypes* are intuitively analogous to powersets: "if A is a type, then Power(A) is the type whose elements are **all** the subtypes of A (including A)" [9]. Based on this notion, MLT defines a *powertype* relation between a higher-order type and a base type at a lower order as follows: *a type t is powertype of a base type t' iff all instances of t specialize t' and all possible specializations of t' are instances of t* (see D3). For example, consider a type called "Person Type" such that all possible specializations of "Person" are instances of it and, conversely, all its instances specialize "Person". In this case, "Person Type" is the powertype of "Person". Since "Person" *is instance of* "1stOT", "Person Type" *is instance of* "2ndOT" and *specializes* "1stOT". Note that it follows from the definition of powertype that "1stOT" *is powertype of* "Individual". Analogously, "2ndOT" *is powertype of* "1stOT", and so on. In other words, the notion of orders or levels in MLT can be seen as a result of the iterated application of Cardelli's notion of powertype to the basic types of MLT.

An important variant of the notion of powertype was discussed by Odell [8]. Odell defined *powertype* simply as a type whose instances are subtypes of another type (the

base type), excluding the *base type* from the set of instances of the *powertype*. Based on Odell's definition for powertypes [8], MLT defines the *characterization* relation between types of adjacent levels: *a type t characterizes a type t' iff all instances of t are proper specializations of t'* (definition D4). The *characterization relation* occurs between a higher-order type t and a base type t' when *the instances of t specialize* t' according to a specific *classification criteria.* Thus, differently from the cases involving (Cardelli's) *is powertype of*, there may be specializations of the base type t' that are not instances of t. For example, we may define a type named "Person Role" (with instances "Employee" and "Client") that *characterizes* "Person", but is not a *powertype of* "Person" since there are specializations of "Person" that are not instances of "Person Role" (e.g. "Child" and "Adult").

Finally, MLT defines some refinements of the cross-level relation of characterization, which are useful to capture further constraints in multi-level models. We consider that *a type t completely characterizes t' iff t characterizes t' and every instance of t' is instance of, at least, an instance of t* (D5). Moreover, *iff t characterizes t' and every instance of t' is instance of, at most, one instance of t* it is said that *t disjointly characterizes t'* (D6). Finally, a common use for the notion of powertype in literature considers a second-order type that, simultaneously, completely and disjointly characterizes a first-order type. To capture this notion MLT defines the *partitions* relation. Thus, *t partitions t' iff each instance of the base type t' is an instance of exactly one instance of t* (D7). For example, we may consider a second-order type "Person Age Phase" (with instances "Child", "Adult" and "Elderly") that *partitions* "Person". A complete formalization of MLT in first-order logic can be found in [13].

Table 1. MLT Axioms

A1	$\forall x\ \text{iof}(x, \text{Individual}) \leftrightarrow \neg\exists y\ \text{iof}(y, x)$
A2	$\forall t\ \text{iof}(t, \text{1stOT}) \leftrightarrow (\exists y\ \text{iof}(y, t) \wedge (\forall x\ \text{iof}(x, t) \rightarrow \text{iof}(x, \text{Individual})))$
A3	$\forall t\ \text{iof}(t, \text{2ndOT}) \leftrightarrow (\exists y\ \text{iof}(y, \text{t}) \wedge (\forall t' \text{iof}(t', t) \rightarrow \text{iof}(t', \text{1stOT})))$
A4	$\forall x\ \big(\text{iof}(x, \text{Individual}) \vee \text{iof}(x, \text{1stOT}) \vee \text{iof}(x, \text{2ndOT})\big) \vee (x = \text{2ndOT})$
D1	$\forall t, t'\ \text{specializes}(t, t') \leftrightarrow (\exists x\ \text{iof}(x, t) \wedge \exists y\ \text{iof}(y, t') \wedge \big(\forall e\ \text{iof}(e, t) \rightarrow \text{iof}(e, t')\big))$
D2	$\forall t, t'\ \text{properSpecializes}(t, t') \leftrightarrow (\text{specializes}(t, t') \wedge t \neq t')$
D3	$\forall t, t'\ \text{isPowertypeOf}(t, t') \leftrightarrow (\exists x\ \text{iof}(x, t) \wedge (\forall t''\ \text{iof}(t'', t) \leftrightarrow \text{specializes}(t'', t)))$
D4	$\forall t, t' \text{characterizes}\ (t, t') \leftrightarrow (\exists x\ \text{iof}(x, t) \wedge (\forall t''\ \text{iof}(t'', t) \rightarrow \text{properSpecializes}(t'', t')))$
D5	$\forall t, t' \text{completelyCharacterizes}(t, t') \leftrightarrow (\text{characterizes}(t, t') \wedge$ $(\forall \text{e}\ \text{iof}(e, t') \rightarrow \exists t''\ (\text{iof}(e, t'') \wedge \text{iof}(t'', t))))$
D6	$\forall t, t' \text{disjointlyCharacterizes}\ (t, t') \leftrightarrow (\text{characterizes}(t, t') \wedge$ $(\forall e, t'', t'''\ (\text{iof}(t'', t) \wedge \text{iof}(t''', t) \wedge \text{iof}(e, t'') \wedge \text{iof}(e, t''')) \rightarrow t'' = t'''))$
D7	$\forall t, t' \text{partitions}(t, t') \leftrightarrow (\text{completelyCharacterizes}(t, t') \wedge \text{disjointlyCharacterizes}(t, t'))$

3 UML's Powertype Pattern Support in a Nutshell

The notion of *generalization set* is central to the UML's powertype pattern support. According to the UML 2.4.1 specification [12] each *generalization set* contains a particular set of *generalizations* that collectively describe the way a specific classifier (a class) is specialized into subclasses. To provide support to the powertype pattern,

UML includes in its "powertypes" package a meta-association that relates a classifier (the so-called "powertype") to a generalization set that is composed by the generalizations that occur between the base classifier and the instances of the powertype [12]. The relation between the powertype and the generalization set is represented in the UML notation by placing the name of the classifier next to the generalization set preceded by a colon. For example, in Fig. 1 three specializations of "Tree" are defined, namely "Elm", "Apricot" and "Saguaro". The text ":Tree Species" denotes that the three subtypes enumerated in the generalization set are instances of "Tree Species" and that "Tree Species" is the "powertype" of the generalization set. Note that the term "powertype" as used in UML does not correspond to the notion of "powertype" as proposed by Cardelli. (This issue is discussed in section 4.) The "disjoint" constraint means that the subtypes have no instances in common while the "incomplete" constraint means that there are instances of "Tree" that are not instances of "Elm", "Apricot" and "Saguaro". The relation between the powertype (e.g. "Tree Species") and the base type (e.g. "Tree") may be represented using a regular association with no special syntax and semantics.

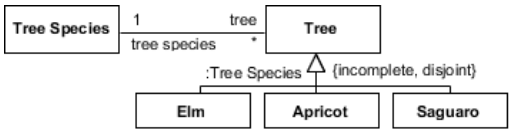

Fig. 1. The UML notation for the powertype pattern (adapted from [12]).

A key observation is that for a classifier to be considered a "powertype" in UML, it must be related to a generalization set. Thus, in UML, the powertype 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 rules out simple models such as one defining "Tree Species" as a "powertype" of "Tree", without forcing the modeler to define specific instances for "Tree Species".

Furthermore, the only syntactic constraint defined in UML concerning powertypes is that "the classifier that maps to a generalization set may neither be a specific nor a general classifier in any of the generalization relationships defined for that generalization set" [12]. While this rule prevents the powertype from being involved in the generalization set defined to represent its own relation with the base type, this constraint is insufficient to rule out scenarios in which the powertype is incorrectly related by generalization with types of any other levels.

4 Applying MLT to Revisit the Powertype Support in UML

The application of MLT to revise the powertype support in UML leads to the formulation of modeling recommendations to ensure: (i) a precise interpretation for the UML constructs used to express the powertype pattern, (ii) a comprehensive support for the powertype pattern including its variants in the literature, and; (iii) a number of syntactic rules to prevent the construction of inconsistent models.

First of all, we should observe that the UML specification is silent with respect to whether Cardelli's notion of powertype can be adopted. However, given that a *gener-*

alization set can be said to define the *classification criteria* used to specialize the general type, the UML notion of powertype seems to correspond to the *characterization relation* in MLT (not to the *is powertype of* relation), in particular as other generalization sets may co-exist defining other classification criteria for the subtypes. This interpretation is corroborated by statements in the specification that explain that the subtypes of a basetype are the instances of the "powertype" (excluding the basetype itself).

Our first recommendation is to mark the association between the base type and the higher order type with the «instantiation» stereotype, in order to distinguish it from other domain relations that do not have an instantiation semantics. An association stereotyped «instantiation» represents that instances of the target type are instantiated by instances of the source type and, thus, denote that there is a *characterization* relation between the involved types (regardless of possible generalization sets). For example, in Fig. 2 an association stereotyped «instantiation» having "Tree" as source and "Tree Species" as target type is used to represent that instances of "Tree" are instances of instances of "Tree Species" and, conversely, that instances of "Tree Species" have instances of "Tree" as instances. Therefore, in MLT terms, it denotes that "Tree Species" *characterizes* "Tree". Since this modeling structure does not rely on generalization sets, the modeler is not forced to represent instances of the powertype, which would have been required in the case of plain UML.

Fig. 2. Illustrating the use of «instantiation».

The multiplicities of the "target" side of an «instantiation» association can be used to distinguish between the different variations of characterization. Whenever the lower bound multiplicity of the target association end is set to one, each instance of the base type is instance of, at least one instance of the powertype. Thus, the higher order type completely characterizes the base type. In contrast, if the lower bound multiplicity of the target association end is set to zero, the inferred characterization relation is not a complete characterization. Analogously, if the upper bound multiplicity of the target association end is set to one, each instance of the base type is instance of, at most one instance of the higher order type. Thus, in this case, the higher order type disjointly characterizes the base type. In contrast, if the upper bound multiplicity of the target association end is set to many (*), the inferred characterization relation is not a disjoint characterization.

Table 2 summarizes the suggested interpretation in terms of MLT, considering different combinations of lower and upper bound multiplicities for the target association end. The combinations of multiplicities of the «instantiation» association with the values of the related generalization set attributes create additional challenges for modelers using the powertype pattern. These combinations are discussed in each of the following subsections, in which we expose some semantic issues.

Lower and upper bound multiplicities set to one. When both the lower and the upper bound multiplicities of an «instantiation» association are set to one, we have

Table 2. The influence of the multiplicities in the semantics of «instantiation» associations.

UML Notation	Semantics in terms of MLT
H «instantiation» (1 ◄ 0..*) B	disjointlyCharacterizes (H, B) ∧ completelyCharacterizes(H, B) ≡ partitions(H, B)
H «instantiation» (0..1 ◄ 0..*) B	disjointlyCharacterizes (H, B) ∧ ¬completelyCharacterizes(H, B)
H «instantiation» (1..* ◄ 0..*) B	completelyCharacterizes(H, B) ∧ ¬disjointlyCharacterizes (H, B)
H «instantiation» (0..* ◄ 0..*) B	characterizes(H, B) ∧ ¬completelyCharacterizes(H, B) ∧ ¬disjointlyCharacterizes(H, B)

that the powertype simultaneously, *completely* and *disjointly characterizes* (i.e. *partitions)* the base type. For example, according to Fig. 2 "Tree Species" *partitions* "Tree" (i.e. each instance of "Tree" is instance of exactly one instance of "Tree Species"). If it is used in tandem with a *complete* generalization set it means that all the instances of the higher-order type are enumerated in the diagram. For example, the model in Fig. 3 (a) represents that: (i) every instance of "Person" must be either an instance of "Man" or an instance of "Woman" and that (ii) "Man" and "Woman" are the only admissible instances of "Person Gender".

At a first superficial inspection, one could consider that «instantiation» associations having the lower bound multiplicity (of the target association end) set to one could only be combined with a *complete* generalization set (as in Fig. 3 (a)). However, this is not the case because the *"complete" constraint* represents whether all instances of the supertype are instances of one of the subtypes *in the generalization set*, and, on its own, it is silent with respect to whether the higher-order type completely characterizes the base type. Thus, a combination of an «instantiation» association having both lower and upper multiplicities set to one in a pattern with an *incomplete* generalization set is admissible, and would mean that there are instances of the higher-order type that are not enumerated in the generalization set. For example, Fig. 3 (b) represents that: (i) each instance of "Tree" is instance of exactly one instance of "Tree Species" (represented by the «instantiation» association), (ii) "Elm", "Apricot" and "Saguaro" are instances of "Tree Species" (represented with the generalization set name), (iii) there are instances of "Tree" that are not instances of "Elm", "Apricot" nor "Saguaro (represented by the *incomplete* constraint). Given the semantics of the «instantiation» stereotype in tandem with the semantics of the incomplete generalization set we can infer that (iv) there are instances of "Tree Species" that are not represented in the diagram.

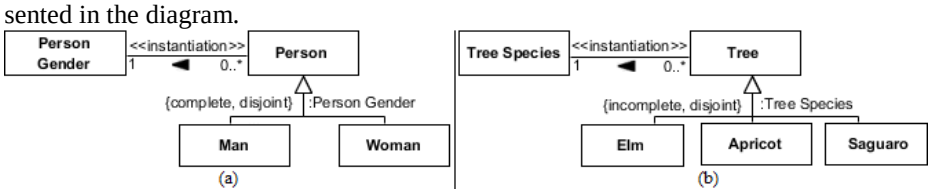

Fig. 3. Using «instantiation» to denote *partitions* relations.

Since the upper bound multiplicity of an «instantiation» association set to one means that each instance of the base type is instance of at most one instance of the

higher-order type, a model combining it in a pattern with an overlapping generalization set is inconsistent, and thus, deemed syntactically invalid.

Lower bound multiplicity set to zero and upper bound set to one. An association stereotyped «instantiation» having the lower multiplicity set to zero and the upper bound multiplicity set to one denotes that the target type *disjointly characterizes* but *does not completely characterize* (in MLT sense) the source type. For example, suppose that an organization defines a type of roles called "Management Role" such that an employee cannot play more than one role of such type and it is not the case that all employees play some "Management Role". This scenario is illustrated in Fig. 4 (a), showing "Organization President" and "Department Dean" as examples of instances of "Management Role". The interpretation of the combination of an «instantiation» association having zero as the lower bound and one as the upper bound multiplicity with an incomplete generalization set is more subtle than the cases we have discussed so far. In order to analyze this combination, we should first note that: (i) there are instances of "Employee" which are not instances of any instance of "Management Role" (as a consequence of the semantics of the «instantiation» association); and (ii) there are instances of "Employee" which are neither "Organization President" nor "Department Dean" (as a consequence of the semantics of *incomplete* generalization sets). The model is still silent with respect to whether all instances of "Management Role" are enumerated in this generalization set. It is possible that there are no other instances of "Management Role", but an interpretation in which there are other management roles not mentioned in the model (e.g. "Division Head") is also admissible.

Since an «instantiation» association having zero as the lower bound multiplicity implies that there are instances of the base type that are not instances of any instance of the higher-order type, a model combining it in a pattern with a complete generalization set is unsound, and thus, is deemed syntactically invalid. Further, as previously discussed, the combination of an «instantiation» association with the upper bound multiplicity set to one in a pattern with an overlapping generalization set is also deemed syntactically invalid.

Lower bound multiplicity set to one and upper bound set to many. An «instantiation» association having the lower multiplicity set to one and the upper bound multiplicity set to "many" (*) denotes that the target type *completely characterizes* but *does not disjointly characterize* (in MLT sense) the source type. For example, suppose that the rules of an organization define a type of roles called "Business Role" (having instances as "Programmer", "DB Designer" and "Sw Designer") such that every employee must play one or more roles of such type.

Associations stereotyped «instantiation» with one as lower bound multiplicity and "many" as upper bound multiplicity can be combined with any generalization sets despite they are *complete or incomplete, disjoint or overlapping*. However, the generalization sets constraints influence the semantics of the diagrams. For example, in the diagram of Fig. 4 (b) the generalization set is *complete* and *disjoint* meaning each instance of "Employee" plays exactly one of the represented instances of "Business Role". Therefore, since the multiplicities of the «instantiation» association between "Business Role" and "Employee" denotes that the instances of the former are over-

lapping, we conclude that there are non-represented instances of "Business Role" such that some of these instances are overlapping between them or some of them are overlapping with the represented ones. If the generalization set of Fig. 4 (b) were defined incomplete we could infer that there were non-represented instances of "Business Role" such that the whole set of instances of "Business Role" classifies all instances of "Employee" having some overlaps. Finally, considering the hypothesis in which the generalization set of Fig. 4 (b) were defined complete and overlapping we would have two possible interpretations: (i) all instances of "Business Role" are represented in the model or (ii) there are non-represented instances of "Business Role" but the represented ones already classify all instances of "Employee" having overlaps between them.

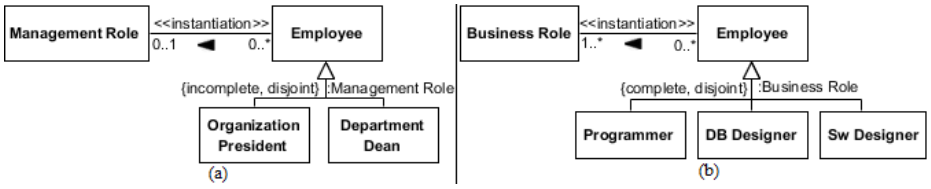

Fig. 4. Using «instantiation» with different multiplicities.

Lower bound multiplicity set to zero and upper bound set to many. An «instantiation» association having the lower multiplicity set to zero and the upper bound multiplicity set to many (*) denotes that the target type *characterizes* (in MLT sense) the source type, however it is neither a *complete characterization* nor a *disjoint characterization*. Therefore, there may be instances of the base type that are instances of more than one instance of the higher-order type and there may be instances of the base type that are not instances of any instance of the higher-order type. For example, Fig. 5 (a) consider a second-order type named "Social Role" whose instances represent roles that instances of "Person" may play in social relations, such as "Client", "Employee" and "Husband". Some instances of "Person" may play more than one "Social Role" and some other instances may play no social role.

Note that it is not possible to infer whether all instances of "Social Role" are represented or not in Fig. 5 (a): (i) they may all be enumerated, or (ii) there may be non-represented instances of "Social Role". If the generalization set of Fig. 5 (a) were *disjoint*, the diagram would still be considered syntactically valid and we could infer that there were non-represented instances of "Social Role" such that the whole set of instances of "Social Role" have some overlaps. Finally, if the generalization set of Fig. 5 (a) were *complete*, the diagram would be considered syntactically invalid since the whole set of instances of "Social Role" does not classify all instances of "Person".

Table 3 summarizes the semantics of the combinations of the multiplicities of «instantiation» associations with the possible constraints of generalization sets, classifying each possible combination as: (i) *enumerated* if it is possible to infer that all instances of the higher-order type are represented in the diagram; (ii) *non enumerated* if one can infer that there are instances of the higher-order type not represented in the diagram; (iii) *silent:* if it is not possible to infer whether the instances of the higher-order type are enumerated or not; or (iv) *invalid* if the combination is syntactically invalid.

Table 3. Analyzing the combination of «instantiation» with generalization set constraints

Association Multiplicities		Generalization sets constraints			
Lower	Upper	{disjoint}		{overlapping}	
		{complete}	{incomplete}	{complete}	{incomplete}
1	1	*enumerated*	*non enumerated*	*invalid*	*invalid*
0	1	*invalid*	*silent*	*invalid*	*invalid*
1	*	*non enumerated*	*non enumerated*	*silent*	*non enumerated*
0	*	*invalid*	*non enumerated*	*invalid*	*silent*

The «powerType» stereotype. Our second recommendation is to use the «powerType» stereotype to represent Cardelli's notion of powertype [9]. If a class stereotyped «powerType» is the target of an «instantiation» association this means that this type *is powertype of* the source type, i.e. the source type and all its specializations are instances of the target element. For example, in Fig. 5 (b), all types that (directly or indirectly) *specialize* "Person" are instances of "Person Type".

According to Cardelli's notion of powertype the base type itself is instance of the higher-order type. Thus, in these cases, the lower bound multiplicity of the «instantiation» association must be set to one and the upper bound to many (*). Moreover, models in which the «powerType» stereotype is applied to types (classifiers) that are not target of any «instantiation» association are deemed syntactically invalid.

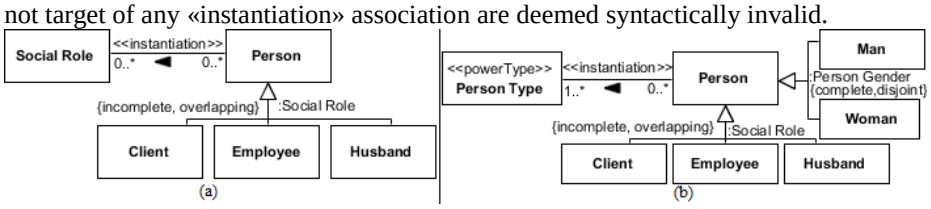

Fig. 5. Using «instantiation» (a) with unbounded multiplicities, and (b) with «powerType».

Another important syntactic constraint involving «powerType» is that, since according to MLT a powertype does not define a classification criteria to be applied to instances of the base type, there should be no generalization set anchored in types stereotyped «powerType» (i.e. *powertype* relations do not give rise to generalization sets). For example, considering the scenario illustrated in Fig. 5 (b), a generalization set named ":Person Type" is not admissible. However, all subtypes of "Person", despite the generalization sets in which they are involved, are instances of "Person Type". Thus, all instances of "Person Gender" and "Social Role" are instances of "Person Type".

Syntactic constraints motivated by MLT rules. An important aspect of the proposed interpretation is that it allows us to leverage the axioms and theorems of the MLT formalization in order to guide the modelers in producing sound models. For instance, given the definition of the *is powertype of* relation of MLT, a type may not have more than one *powertype* and a higher order *type* may be a *powertype of* at most one other type. This suggests a clear syntactic constraint: a class stereotyped «powerType» can only be target of at most one «instantiation» association and a regular class can only be the source of at most one «instantiation» association having as target a class stereotyped «powerType».

The MLT theorem stating that if a type *t specializes* a type *t'* then the *powertype of t specializes* the *powertype of t'* may be used to check the syntax of powertype hierarchies, and also to generate the powertypes hierarchy corresponding to the base types hierarchy. For example, in Fig. 6 (a) the conjunction of the facts that: (i) "Employee *specializes* "Person", (ii) "Person Type" *is powertype of* "Person" and (iii) "Employee Type" *is powertype of* "Employee" implies that "Employee Type" must *specialize* "Person Type".

Considering the MLT definitions of *powertype*, *characterization* and *proper specialization* we conclude that if a type *t'* is *powertype of* a type *t* and a type *t''* *characterizes* the same base type *t* then all instances of *t''* are also *instances of t'* and, thus, *t'' proper specializes t'*. This theorem also suggests a syntactic constraint. For example, in Fig. 6 (a) "Management Role" *characterizes* "Employee" and *specializes* "Employee Type", whereas "Person Gender" *characterizes* "Person" and *specializes* "Person Type". In this case, if the modeler fails to include any of the specializations between the higher-order types, it would be possible to infer them automatically.

Another MLT theorem states that if two types *t'* and *t''* both *partition* the same type *t* then it is not possible for *t'* to specialize *t''*. Again this suggests a clear syntactic constraint. For example, in Fig. 6 (b), "Person Age Phase" *partitions* "Person" according to their age having "Child" and "Adult" (and other non-represented types) as instances. "Person Gender", in turn, *partitions* "Person" according to their gender having "Man" and "Woman" as instances. Thus, to be syntactically valid, the model may not include a specialization between "Person Age Phase" and "Person Gender".

Recall that the MLT cross-level relations (*characterization* and *is powertype of*) hold between a higher-order type and another type at one order lower. Thus, if two types are linked through an «instantiation» association the type at the source association end is at an order lower than the one in the target (e.g. in Fig. 6 (b) "Person" is one order lower than "Person Age Phase"). Hence, cycles of associations stereotyped «instantiation» are not allowed. For example, suppose *A* is the target in an «instantiation» association in which *B* is the source while *B* is the target in another «instantiation» association in which *A* is the source. This scenario is absurd since *A* must be at one order lower than *B* and, simultaneously, *B* must be at one order lower than *A*.

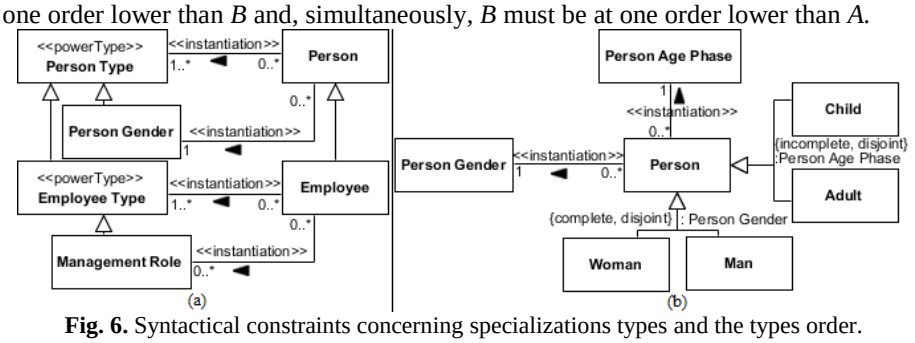

Fig. 6. Syntactical constraints concerning specializations types and the types order.

Finally, we consider that all higher-order types represented in diagrams must have cross-level relations with other types. Thus, we can determine the order of a type considering the «instantiation» associations in which they are involved as target. Types that are not targets of any «instantiation» association are first-order types (e.g.

"Person", "Man", "Woman", "Adult" and "Child" in Fig. 6 (b)). Types that are target in «instantiation» associations in which the sources are first-order types are second-order types (e.g. "Person Gender" and "Person Age Phase" in Fig. 6 (b)), and so on. The MLT axiom that states that each domain type must be instance of exactly one MLT basic type (being thus at only one order) can be syntactically verified in our models. Further, the MLT theorem that specialization relations may only hold between two types at the same order may also be syntactically verified. For example, in Fig. 6 (b) there may not be specialization relations between a first-order type (i.e., "Person", "Man", "Woman", "Adult" or "Child") and a second-order type (i.e. "Person Gender" or "Person Age Phase"). Otherwise, the model would be considered syntactically invalid. A prototype plugin for the Visual Paradigm modeling tool that implements the proposed profile and performs syntactic verification of MLT rules is available at http://github.com/nemo-ufes/MLT-VP-plugin.

5 Related Work

An early attempt to address multi-level modeling by Odell [8] defined the concept of powertype informally using regular associations between the powertype and a base type. This differs from our approach because we use constructs having specialized semantics to denote the cross-level relations between types defined in MLT. This allows us to prescribe syntactic rules for the models that use these relations following the axioms in the formal theory.

Similarly to Odell [8], Gonzalez-Perez and Henderson-Sellers [7] use an association labeled "partitions" between a powertype and a base type (called a "partitioned type" in their terminology). The authors illustrate their technique with a diagram in which "partitions" is modeled as a many-to-one 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 lower bound multiplicity of the "partitions" association in the "TaskKind" end cannot be *one*. 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). This is an example of a mistake, which could be avoided with a richer language support for the powertype pattern and its variants, as we propose here.

The concept of powertype is founded on the notion that "instances of types can also be types" [8]. Motivated by a similar observation, Atkinson and Kühne [17] defined the notion of *clabject*, which is valuable to our approach. They discuss that every instantiable entity has both a type (or class) facet and an instance (or object) facet. In our approach, instances of higher-order types may be considered *clabjects*. For instance, considering the previous example all instances of "TaskKind" as well as all instances of "TaskPowertype" have their own instances being, thus, *clabjects*.

Atkinson and Kühne have also proposed a *deep instantiation based* approach [6], [19] 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 authors consider the main benefit of deep instantiation is to support multi-level modeling

without the need of introducing the required base type in the powertype pattern, which they consider superfluous [19]. For example, using this approach it is possible to define mobile phone models, such as "IPhone6" and "GalaxyS6", omitting the notion of "Mobile Phone" from the domain model. Important consequences of omitting base types are that the modeler become unable to express whether the instances of a higher-order type (mobile phone model in this example) are disjoint and/or covering types and we are also prevented from determining metaproperties (such as e.g., rigidity) of the base type (mobile phone in this case). It is worth noticing that the deep instantiation approach allows the modeler to represent the base type if it is deemed desirable. However, if the modeler decides to represent the base type, the approach does not provide constructs to represent the relation between it and the higher-order type, not distinguishing thus between the different possible kinds of cross-level relations. As a consequence, the approach does not provide mechanisms to check if the rules concerning these relations are respected, e.g., to guarantee that all instances of the higher-order type ("Mobile Phone Model") specialize the base type ("Mobile Phone").

Telos [20] is a knowledge representation language that supports the representation of types having other types as instances (i.e. *clabjects*). Roughly 30 axioms are defined to formalize Telos' principles for instantiation, specialization, object naming and attribute definition [20]. Although it supports multi-level modeling through its notion of type, it does not elaborate on the nature of cross-level relations between higher-order types and base types. Further, it does not employ systematically the powertype pattern, although we consider it would be possible to extend the Telos built-in support by using its features of user-defined constraints and rules to formally define the cross-level structural relations proposed in MLT.

6 Final Considerations

In this paper, we have addressed multi-level modeling from the perspective of the *powertype pattern*. We have used a well-founded reference theory to support the analysis and revision of the powertype support, demonstrating that the current support lacks *expressivity*, *clarity*, and *parsimony*.

By employing the result of this analysis, we propose a lightweight extension of the UML to address the exposed limitations. We use the formal rules inherent to MLT to systematically incorporate syntactic constraints in the profile thus guiding the modeler to produce sound multi-level models. Our approach is able to distinguish properties of the relation between higher-order and base types that cannot be expressed in UML and that are required to represent multi-level classification schemes.

In [3], one of us has evaluated a fragment of UML at light of the Unified Foundational Ontology (UFO). Based on this analysis, a UML extension for the purposes of conceptual modeling (dubbed OntoUML) has been proposed. The ontology was used as a theory to inform the definition of a profile with syntactic constraints that reflect the UFO axioms. In this paper, we have applied a similar approach to extend UML class diagrams using MLT as a theory to incorporate distinctions and constraints for multi-level modeling. In [15], we have already combined MLT and UFO in order to leverage both benefits of the foundational ontology and the multi-level modeling theory. A natural extension of this work is to enrich OntoUML with the support for the powertype pattern as discussed here.

Finally, we aim at applying MLT to analyze and enrich the semantics of the so-called deep modeling approaches [5, 19]. An ongoing effort is the extension of Atkinson and Kühne's approach [19] to support MLT's cross-level relations, which should combine the benefits of deep instantiation and the powertype pattern.

Acknowledgments. This research is funded by the Brazilian Research Funding Agencies CNPq (grants number 311313/2014-0, 485368/2013-7, 312158/2015-7 and 461777/2014-2) and CAPES. The authors would like to thank Claudenir M. Fonseca for implementing the Visual Paradigm plugin for the UML profile presented here.

References

1. Halpin, T., Morgan, T.: Information Modeling and Relational Databases. Morgan Kaufmann, San Francisco (2008)
2. Recker, J., Rosemann, M., Green, P., Indulska, M.: Do Ontological Deficiencies in Modeling Grammars Matter?. MIS Quarterly, Vol. 35, No. 1, 57-79 (2011)
3. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models. University of Twente, Enschede, The Netherlands (2005)
4. Mayr, E.: The Growth of Biological Thought: Diversity, Evolution, and Inheritance. Belknap Press, Cambridge (1982)
5. 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, pp. 107-116 (2009)
6. Atkinson, C., Kühne, T.: The Essence of Multilevel Modeling. In: Proc. Of the 4th International Conference on the Unified Modeling Language, pp. 19-33. Toronto, Canada (2001)
7. Gonzalez-Perez, C., Henderson-Sellers, B.: A powertype-based metamodeling framework. Software & Systems. Modeling, 5(1), pp. 72-90 (2006)
8. Odell, J.: Powertypes. In: Journal of Object-Oriented Programing, 7(2), pp. 8-12.(1994)
9. Cardelli, L.: Structural Subtyping and the Notion of Powertype. In Proc. Of the 15th ACM Symposium of Principles of Programming Languages, pp. 70-79 (1988)
10. ISO/IEC.: ISO/IEC 24744: Software Engineering –Metamodel for Development Methodologies. ISO, Geneva (2007)
11. Fowler, M.: Analysis Patterns: Reusable Object Models. Addison-Wesley (1997)
12. OMG: UML Superstructure Specification – Version 2.4.1. (2011)
13. Carvalho, V. A., Almeida, J. P. A.: Towards a Well-Founded Theory for Multi-Level Conceptual Modeling. Submitted (2015) available at http://nemo.inf.ufes.br/mlt
14. Carvalho, V. A., Almeida, J. P. A.: A Semantic Foundation for Organizational Structures: A Multi-level Approach. In: 19th IEEE Intl Enterprise Distributed Object Computing Conference (EDOC 2015), pp. 50-59. Adelaide, Australia (2015).
15. 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: 35th Intl. Conf. on Conceptual Modeling (ER 2015), pp. 119-133 (2015)
16. Guizzardi, G. et al.: Towards Ontological Foundation for Conceptual Modeling: The Unified Foundational Ontology (UFO) Story. Applied Ontology, Vol. 10, issues 3-4, IOS Press (2015).
17. C Atkinson, C., Kühne, T.: Meta-level Independent Modeling. In: Intl Workshop "Model Engineering" (in conj. with ECOOP'2000). Cannes, France (2000)
18. Pirotte, A., Zimanyi, E., Massart, D., Yakusheva, T.: Materialization: a powerful and ubiquitous abstraction pattern. In: 20 th Intl. Conf. Very Large DataBases, pp. 630–641 (1994)
19. Atkinson, C., Kühne, T.: Reducing accidental complexity in domain models. Software & Systems Modeling, 7(3), pp 345-359. Springer-Verlag (2008)
20. Jeusfeld, M. A.: Metamodeling and Method Engineering with ConceptBase. In: Jeusfeld, M. A., Jarke, M., Mylopoulos, J. (Eds.), Metamodeling for Method Engineering, pp 89-168. The MIT Press (2009)