Incorporating Types of Types in Ontology-Driven Conceptual Modeling

Claudenir M. Fonseca[1], Giancarlo Guizzardi[1,2], João Paulo A. Almeida[3], Tiago Prince Sales[1], and Daniele Porello[4]

[1] Conceptual and Cognitive Modeling Research Group, Free University of Bozen-Bolzano, Italy
{gguizzardi,cmoraisfonseca,tprincesales}@unibz.it
[2] Services & Cybersecurity Group, University of Twente, The Netherlands
[3] Ontology & Conceptual Modeling Research Group, Federal Univ. of Espírito Santo, Brazil
jpalmeida@ieee.org
[4] University of Genoa, Italy
daniele.porello@unige.it

Abstract The Unified Foundational Ontology (UFO) has been used to provide foundations for the major conceptual modeling constructs. So far, UFO has reflected a view in which domain entities are fundamentally divided into those that collect invariants of the domain (i.e., types) and those entities that manifest those invariants (i.e., instances), following the conventional two-level classification scheme. This paper extends UFO with support for multi-level classification schemes, in which some entities accumulate both type-like and instance-like characteristics. This requires an ontological interpretation and a formal theory of types of types. This theory is employed to engineer new constructs and constraints into the OntoUML language, and to develop computational support for the formal verification of constraint violation over multi-level conceptual models.

Keywords: ontology-driven conceptual modeling · multi-level modeling · high-order types · UFO · OntoUML

1 Introduction

The Unified Foundational Ontology (UFO) has been used to provide foundations for the major conceptual modeling constructs [19, 23]. This ontology has led to the OntoUML Ontology-Driven Conceptual Modeling language [19, 21], a UML class diagram profile reflecting the ontological micro-theories comprising UFO. Despite the increasing adoption [36] and successful application [23] of this language to address problems in a variety of areas, so far, UFO's foundations to OntoUML have reflected a view in which domain entities are fundamentally divided into those that collect invariants of the domain (i.e., types) and the entities that manifest those invariants (i.e., instances). However, in a multitude of domains, entities that accumulate both type-like and instance-like characteristics play a central role. For example, in biological taxonomy, the type *Canis familiaris* collects the properties manifested in individual dogs (e.g., *having fur*, *being quadrupeds*) while manifesting itself properties of the type Species (e.g., *having a certain biological origin*, *having an expected lifespan*). In other words, in this domain,

we are concerned not only with types of individuals (i.e., *first-order* types), but also with types of types (i.e., *high-order* types). Models that extend the traditional two-level scheme by accommodating types that are themselves instances of other (high-order) types are called multi-level models [5].

This paper extends UFO and then OntoUML with support for multi-level modeling, including an ontological interpretation and formal model of high-order types. By leveraging a formal theory called MLT (Multi-Level Theory) [9], we incorporate a micro-theory of high-order types in UFO. This theory is then employed to engineer new constructs and constraints into the OntoUML language and to develop computational support for the formal verification of constraint violation over these models. We position our work with respect to competing approaches present in the literature and demonstrate how our contributions answer an existing demand in conceptual modeling for ontologically sound and semantically precise support for types of types.

A distinctive feature of our approach is that, by considering types as *endurants*, we can account for qualitative changes that these types may undergo in time [20]. This means that the UFO ontological categories applicable to types of individual endurants (such as kinds, phases, roles, etc.) [22] also apply to types of types. For instance, in biology, this allows us to account for high-order types such as Endangered Species or Extinct Species as *phases*, as the very same species can instantiate these types in different situations. Treating types as endurants also allows us to account for temporal properties of their existence: this is particularly important for types such as social roles, artifactual types, and nominal kinds in general [35]. For example, consider the numerous crime types defined in most penal codes; these come into existence at determinate times as a result of the exercise of legislative institutional power. The same move also allows us to account for contingent or accidental properties of high-order types: for example, an animal species may be characterized by a changing population size; a car model may be characterized by a recommended sales price.

Empirical evidence for high-order types shows their relevance: a study of Wikidata in 2016 encountered over 17,000 classes involved in multi-level taxonomies [6]. That study already indicated the importance of rules to detect problematic usage of high-order types in that knowledge base, which was corroborated by an updated study in 2021 [12]. The latter found over 5 million problematic statements in Wikidata that could have been avoided with automated rules. The inconsistencies found over the years reveal that modelers struggle to represent high-order types adequately [6, 12]. In OntoUML, there are 123 documented occurrences of high-order types across 19 ontologies in a dataset of 113 ontologies.[5] This count considers classes explicitly labeled as types (e.g., Person Type) and those decorated with various *ad hoc* stereotypes (e.g., «type», «powertype», and «high-order type»). This indicates that modelers *subvert* [23] the modeling language in order to capture somehow types of types even when support for them is not explicitly part of the language, a phenomenon that has also been observed in OntoUML with event types [2] and with reified properties [22].

The remainder of this paper is structured as follows: Section 2 presents the UFO and OntoUML background, discussing why the current status lacks full support for high-order types; Section 3 introduces high-order types explicitly in UFO; Section 4

[5] Dataset available at `https://purl.org/ontouml-models/git`

presents the corresponding OntoUML extension; Section 5 discusses related work and Section 6 presents our final considerations.

2 Background and Motivation

OntoUML is an ontology-driven conceptual modeling (ODCM) language that focus on the representation of types of object-like entities termed *endurants*, and their type-level relations. In UFO, *endurant types* (Fig. 1) are types of entities that exist in time and that can undergo changes while keeping their identity. Endurants encompass (i) *existentially independent* objects such as a person, an organization, or a car, whose types are called Substantial Types and (ii) *existentially dependent* aspects of objects, such as Paul's height, or John's employment at Big Tech Inc., whose types are called Moment Types. The *existential dependence* relation tying moments to their bearers is called *inherence*.

Moment types are further classified into types of intrinsic (internal) moments, Quality Type and Intrinsic Mode Type, and types of extrinsic (relational) moments, Relator Type and Extrinsic Mode Type. Quality types classify aspects of endurants that can be mapped to some quality space, such as a person's height, or a car's color; intrinsic mode types, on the other hand, classify internal aspects of endurants that are not conceptualized in terms of a quality space, such as one's ability to speak English; extrinsic mode types classify relational aspects that simultaneously inhere on a unique endurant and *externally depend* on another. Extrinsic modes ground unilateral relations such as *John loves Mary*; finally, relator types classify relational aspects of endurants that inhere in the sum of all endurants it involves, such as John's employment at Big Tech Inc., which is the sum of aspects of John as the employee as well as aspects of Big Tech Inc. as the employer.

The aforementioned classification forms a taxonomy of endurant types based on the ontological nature of their instances. On top of that, endurant types in UFO are also

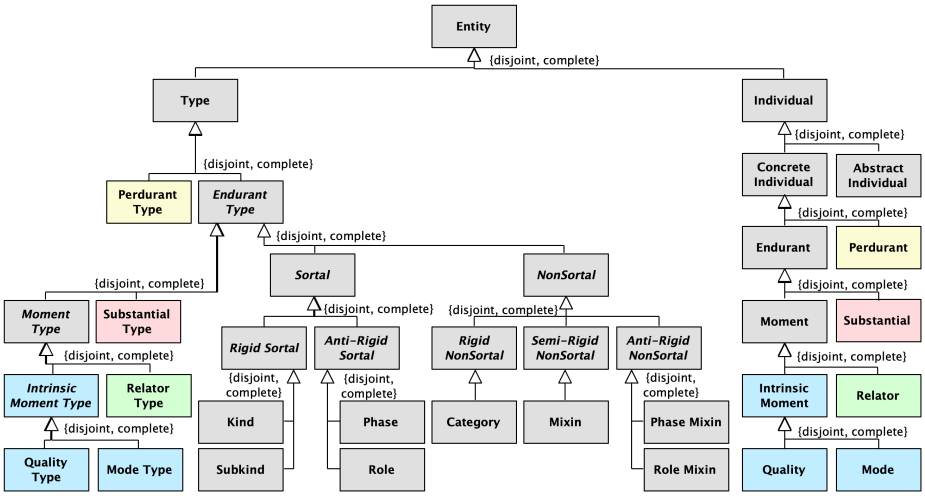

Figure 1: UFO's original taxonomy of endurant types.

classified according to orthogonal characteristics of how they apply to their instances. Sortals are endurant types that provide a uniform *principle of identity* for their instances, i.e., a principle capturing what are properties that two instances of that type must have in common in other for them to be the same. In particular, the identity principle informs which changes an endurant may undergo while preserving its identity. A sortal can either directly provide the identity principle for its instances, or inherit that principle provided by another sortal. Non-sortal types, on the other hand, classify endurants that follow distinct identity principles, being also referred to as *dispersive types* [19]. Non-sortals aggregate properties that are common to different sortals. Therefore, while Computer, Mobile Phone, and Tablet are examples of sortals following distinct identity principles (i.e., their identity principles are clearly based on distinct sets of properties), Electronic Device is an example of non-sortal that aggregates properties that are common to the instances of each of these three sortals, among others.

Sortals and non-sortals are further differentiated according to their rigidity. Rigid types necessarily classify (in the modal sense) endurants, i.e., a rigid type classifies its instances as long as they exist. Anti-rigid types, in contrast, classify their instances contingently, having their instances moving in and out of their extension without ceasing to exist, i.e., while maintaining their identities. For example, Person is typically conceived as a rigid type as, in most conceptualizations, it classifies its instances at all times. In contrast, Student and Teenager only classify persons manifesting certain (accidental) properties, namely *holding an enrollment at an educational institution* and *having an age between thirteen and nineteen years old*. It is also possible that types necessarily classify some of its instances but not others, being referred as semi-rigid types. An example of semi-rigid type is Insured Item which accounts for domains in which some entities are necessarily insured (e.g., cars), while others are not (e.g., bikes, cellphones). UFO only accounts for semi-rigid types among non-sortals, which are called Mixins.

All sortal types sharing the same identity principle inherit it from a unique sortal a common *ultimate sortal*, or Kind. For this reason, kinds rigid types sitting atop sortal taxonomies, *necessarily* classifying the entities that obey the identity principle it supplies. Other rigid endurants types are classified as either Subkind or Category, depending on whether their instances conform to a single unique identity principle or not, respectively.

Anti-rigid types grounded on relational (i.e., extrinsic) properties of their instances are referred to as roles, being instances of Role or Role Mixin, depending on whether their instances conform to a single principle of identity. The same sortal differentiation applies to phases, which are classified into Phase and Phase Mixin. Phases are grounded on variances of internal (i.e., intrinsic) properties of their instances.

OntoUML combines the leaf categories of these orthogonal taxonomies of endurant types in Fig. 1 in order to define its constructs. For example, we can have substantial kinds, substantial phases, relator kinds, relator phases, quality subkinds, etc. Stereotypes in a UML profile decorate classes and associations providing the specialized UFO semantics. The constructs of OntoUML are governed by a number of *semantically-motivated syntactical constraints* that ensure that any valid OntoUML model represents a sound conceptualization in terms of UFO. These constraints are implemented in tools that automatically detect errors in a conceptual model [22]. For example, a specific constraint rules out a model in which a phase is specialized by a subkind; this model

would be unsound by definition, as an anti-rigid type (e.g., a phase) cannot be a supertype of a rigid type (e.g., a subkind). For a detailed account of the various rules derived from the UFO taxonomy of endurant types, we refer the reader to [22].

An example of an OntoUML model in the shipping domain is shown in Fig. 2. This model considers that a Ship can go through different phases (Docked Ship, On Route, Decommissioned), can be of different subkinds (Panama-class Ship, or Suez-class Ship). The use of the «kind» stereotype indicates that ships and persons are substantials. Ships are characterized here by «quality» Size, instances of which are qualities inhering in a particular ship and representing its length, weight and hull draft. The model also exemplifies the Assignment of a ship to a Person contingently playing the role of Captain. The stereotype «relator» indicates Assignment is a relator kind. Since relators are endurants, they can also be classified using the taxonomy of endurant types [22]. Here, assignment relators are classified into Temporary and Permanent assignment, which are considered to be phases, reflecting a domain rule that a temporary assignment may become permanent (e.g., after a certain duration).

Note that all types in Fig. 2 capture invariants applicable to individuals: substantials (persons, ships), their qualities (ships' sizes) and relators (assignments). However, we may also be interested in capturing various domain-specific aspects of types themselves. For example, consider when *ship types themselves* become relevant for modeling aspects of this domain. Some ship types (such as Panama-class Ship, or Suez-class Ship) are *created by* regulations, are *applicable to* certain canals (Panama, Suez), and establish a *maximum size* for the ships that instantiate these classes. Persons may be *licensed to captain* specific *ship types*; we may be interested in knowing the current *number of ships* of a certain *ship type* that are on route, whether a *ship type* is *under production* or *discontinued*, etc. Further, we can anticipate in this domain that the model is not exhaustive with respect to ship types; thus, the focus shifts to invariant aspects of ship types, that we may have in the future (not unlike particular ships and persons, these do not have to be 'hard-coded' in the model). Thus, we need to introduce the notion of Ship Type in our model, itself a high-order type (or metatype) as its instances are types. However, the UFO taxonomy shown in Fig. 1 and reflected in OntoUML does not confer to *types* the same possibilities that are conferred to endurant individuals. In the following sections, we refactor the UFO taxonomy shown earlier in order to ascribe full endurant status to types, later revisiting OntoUML's stereotypes and rules to support high-order types. We also incorporate the MLT theory [1, 10] as a microtheory of UFO in order to account for multi-level phenomena (such as the notion of type *order*).

3 Extending UFO with High-Order Domain Endurant Types

In order to accommodate high-order types in UFO's domain of inquiry, we rely on a notion of instantiation (iof for short) that breaks away with the two-level divide and admits that types may classify not only individuals, but other types as well. Hence, we incorporate into UFO the notion of instantiation provided by MLT, where iof is a primitive relation that holds between an instance e and a type t in a world w where t classifies e (a1). UFO entities can thus be divided into those that can possibly have

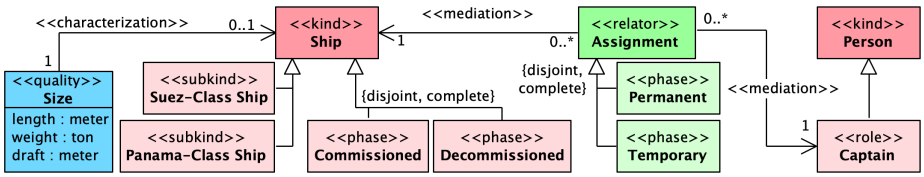

Figure 2: Example OntoUML model in the domain of maritime ships.

instances (in a modal sense) (a2) and those that cannot (a3), i.e., between types and individuals (see Fig. 4).

a1 $\mathsf{iof}(e,t,w) \rightarrow \mathsf{entity}(e) \wedge \mathsf{type}(t) \wedge \mathsf{world}(w)$

a2 $\mathsf{type}(t) \leftrightarrow \mathsf{entity}(t) \wedge \exists e,w\,(\mathsf{iof}(e,t,w))$

a3 $\mathsf{individual}(i) \leftrightarrow \mathsf{entity}(t) \wedge \neg\mathsf{type}(i)$

This basic distinction of entities allows for the characterization of three fundamental classification schemes: (i) the two-level scheme, which include individuals and types of individuals, also referred to as *first-order types*; (ii) the strictly stratified scheme, which extends the two-level scheme by including *second-order types* (i.e., types of first-order types), *third-order types* (i.e., types of second-order types), and so on; and (iii) the non-stratified schemes, where types' extensions span across the boundaries of any particular order. For example, consider the entities in the domain represented schematically in Fig. 3 where all dashed arrows represent iof relations. At the lowest level of classification, we have two individuals (particular animals) Chilly Willy, instantiating Emperor Penguin, and Lassie, instantiating Dog and Collie. These five entities fit the two-level scheme of individuals and first-order types. Moving higher in the multi-level classification scheme, Species and Breed are examples of types classifying first-order types, i.e., second-order types, where the former classifies Emperor Penguin and Dog, while the latter classifies Collie. Likewise, the instantiation chain continues with Taxonomic Rank, a third-order type, classifying Species and Breed. All of these types are instances of Biological Concept. The latter (with instantiation spanning across various levels indicated by red dashed arrows) is an example of an *orderless type*.

Note that the adopted characterization of types allows the stratified scheme to extend with no upper boundary, accounting for instantiation chains as long as necessary for any

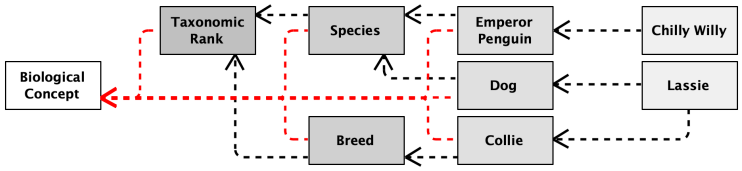

Figure 3: Multi-level classification in the biology domain.

particular domain. Moreover, we refer to the types sitting beyond the first-order, or not bound to a particular order (i.e., orderless), as *high-order types*.

As discussed earlier, endurants in UFO are object-like entities that exist *in time*, that are the subject of change, and that maintain identity throughout such changes. These are characteristics that we also confer here types (as motivated earlier with the maritime example). Hence, we must refactor the taxonomy by introducing Endurant as a more abstract notion, subsuming Type and the original notion of Endurant, now adequately renamed Endurant Individual, see Fig. 4.

In multi-level models, a set of relations that we refer here as *structural relations*, characterize how types are related in terms of their intensions, i.e., the properties they collect about their instances. The most fundamental of these is the specialization relation, which we differentiate here between specialization (a4) and proper specialization (a5). A type t_1 specializes a type t_2 iff every possible instance of the former is necessarily instance the latter; t_1 specializes t_2 iff t_1's intension includes t_2's. Since every type includes its own intension, specializes is a reflexive relation. The proper specialization relation, on the other hand, characterizes the specialization between two different types. Specialization relations, proper or otherwise, can only hold when the specialized type (or *supertype*) is an orderless type or an ordered type of the same order (or level) as the specializing one (or *subtype*) [1].

a4 $\mathsf{specializes}(t_1, t_2) \leftrightarrow \mathsf{type}(t_1) \wedge \mathsf{type}(t_2) \wedge \forall e, w\,(\mathsf{iof}(e, t_1, w) \rightarrow \mathsf{iof}(e, t_2, w))$

a5 $\mathsf{properSpecializes}(t_1, t_2) \leftrightarrow \mathsf{specializes}(t_1, t_2) \wedge (t_1 \neq t_2)$

Other two structural relations emerge from the *powertype* variants as defined by Cardelli [7] and Odell [30]. These are structural relations that hold from a type of a

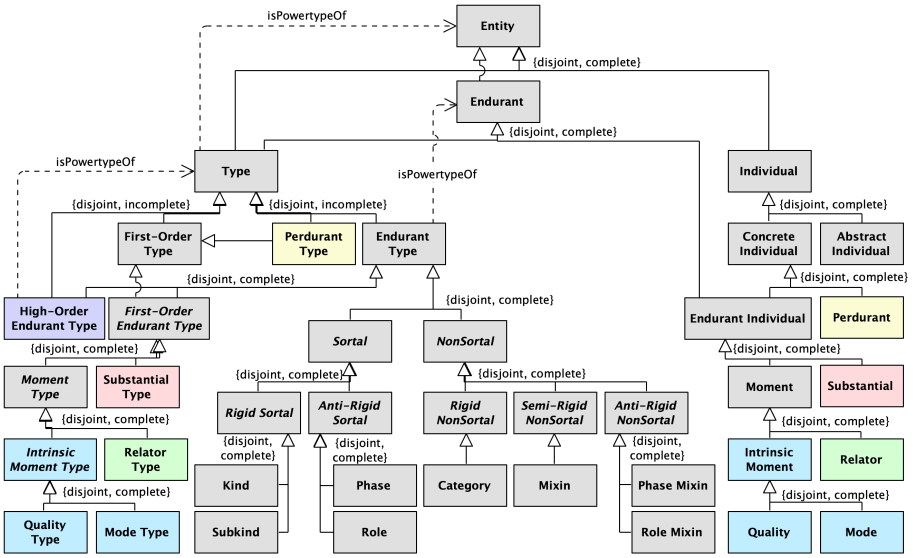

Figure 4: UFO extension for high-order endurant types.

higher order to a *base type* of the order immediately below (or between two orderless types). In UFO, a type t_1 is powertype of a base type t_2 iff every specialization of the latter is an instance of the former (a6), following Cardelli's notion. From Odell's notion, a type t_1 categorizes a base type t_2 iff every instance of the former is a proper specialization of the latter. In other words, the instances of a (Cardelli) powertype are all those whose intensions include a base type's intension; in turn, the instances of a categorizer (following Odell) are those types whose intensions include not only a base type's intension, but also some further restrictions defined in that categorizer.

a6 $\mathsf{isPowertypeOf}(t_1, t_2) \leftrightarrow \mathsf{type}(t_1) \wedge \mathsf{type}(t_2) \wedge \forall t_3, w\,(\mathsf{specializes}(t_3, t_2) \leftrightarrow \mathsf{iof}(t_3, t_1, w))$

a7 $\mathsf{categorizes}(t_1, t_2) \leftrightarrow \mathsf{type}(t_1) \wedge \mathsf{type}(t_2) \wedge \forall t_3, w\,(\mathsf{iof}(t_3, t_1, w) \rightarrow \mathsf{properSpecializes}(t_3, t_2))$

In the original UFO taxonomy of Fig. 1, every type in the taxonomy Individual, with the exceptions of Concrete Individual and Abstract Individual, has a powertype in the taxonomy of Type: Type is the powertype of Individual, Endurant Type is the powertype of Endurant, an so on. Moreover, the taxonomy of the types in the taxonomy of Individual are first-order types, while the types of the taxonomy of Type are second-order types.

With the extension of Fig. 4 this is no longer the case as the taxonomy of Type classifies types of any order (and Type is thus orderless). The mirrored taxonomies seen previously are still present in this extension, given the proper adjustments: First-Order Perdurant Type, First-Order Endurant Type, and the specializations of it are the powertypes of Perdurant, Endurant, and its specializations. However, important changes emerge at the top of UFO's taxonomy. Type becomes the powertype of Entity (following MLT) since every type that classifies entities (i.e., every specialization of Entity) is instance of Type. For the same reason, Endurant Type becomes the powertype of the newly introduced Endurant. High-Order Endurant Type is the powertype of Type. The types that we can now admit with this extension are instances of High-Order Endurant Type: endurant types whose instances are other types (endurant types themselves or not).

4 Extending OntoUML with Support for High-Order Types

OntoUML is a *lightweight extension* of the Unified Modeling Language (UML) [29] designed to support ODCM. Through UML's profiling mechanism, OntoUML defines a collection of class- and association stereotypes that reflect the ontological distinctions present in UFO. OntoUML also defines a set of *semantically-motivated syntactical constraints* that govern how the language's constructs can be employed and ensures that every syntactically correct model represents a sound UFO-based ontology. The OntoUML profile presented here is implemented as in a UML CASE tool which includes the automated verification of syntactical constraints in users' models[6].

[6] The OntoUML plugin for Visual Paradigm is available at `https://purl.org/krdb-core/ontouml-plugin`.

4.1 Stereotypes and tagged values

Fig. 5 presents the extended OntoUML profile, enabling the representation of high-order endurant types (we use the term 'classes' here following UML convention). Stereotypes of endurant classes are used in a model to position them under the two orthogonal taxonomies of Endurant Type of UFO (cf. Fig. 4). One of these taxonomies settles the sortality, rigidity and external dependence of types, and the other settles the ontological nature of a type's instances (i.e., whether the instances of a type are objects, relators, modes qualities or other types). This is achieved by three complementary design choices: (i) through the representation of the variations in sortality, rigidity and external dependence as stereotypes (concrete class stereotypes in gray in the diagram, namely, «subkind», «role», «phase», «category», «mixin», «roleMixin», «phaseMixin»); (ii) through the differentiation of kinds (i.e., ultimate sortals) based on unique ontological natures (the concrete stereotypes: «type» for high-order kinds, «kind» for substantial kinds, «relator» for relator kinds, «quality» for quality kinds and «mode» for mode kinds); and (iii) through the use of the restrictedTo *tagged value* that determines the possible ontological natures of the entities present in the extension of the decorated class (type, object – as a synonym for substantial –, relator, mode and quality). The latter tagged value is automatically derived when a class specializes an ultimate sortal, as the ultimate sortal settles the ontological nature of its instances. A suggested color scheme reflects these ontological natures, and is adopted in the examples in this paper (classes of types in purple, of objects in red, of relators in green and of intrisic moments in blue). The stereotypes in white reflect the taxonomic structure of UFO, they are abstract, and as such they are not directly available to the modeler.

Generalization sets may be used to indicate which higher-order types are instantiated in subclasses in line with plain UML. The «instantiation» stereotype is used to provide specialized semantics to an association between a higher-order type and a base type following [8], a solution which, differently from plain UML, can cater for the cases in which no explicit specializations of the base type are provided. When the higher-order type is tagged isPowertype=true, it is a Cardelli powertype of the base type; otherwise, it is a categorizer of the base type. An order tagged value is also included (an is typed UnlimitedNatural to allow for orderless types).

4.2 Revisiting the ship domain with the extended profile

We now revisit the maritime ship example with the OntoUML diagram shown in Fig. 6. We introduce a Ship Type powertype (of order 2) for Ship, with a number of specializations: Licensed Ship Type is the role played by ship types in the context of a Captain License, ship types can be Discontinued (obsolete) or Current, a subkind of Ship Type, namely, Ship Class by Size classifies ship types using maximum size for those classes. The generalization set annotation indicates that Suez-Class Ship and Panama-Class Ship are instances of Ship Class by Size, and thus can be related to a Canal (e.g., the Panama Canal, the Suez Canal) through the establishment of Canal Restrictions that determine the admissible ship classes for that canal. (The orders of subclasses are inherited from their superclasses and can be omitted here since order is defined for Ship Type.)

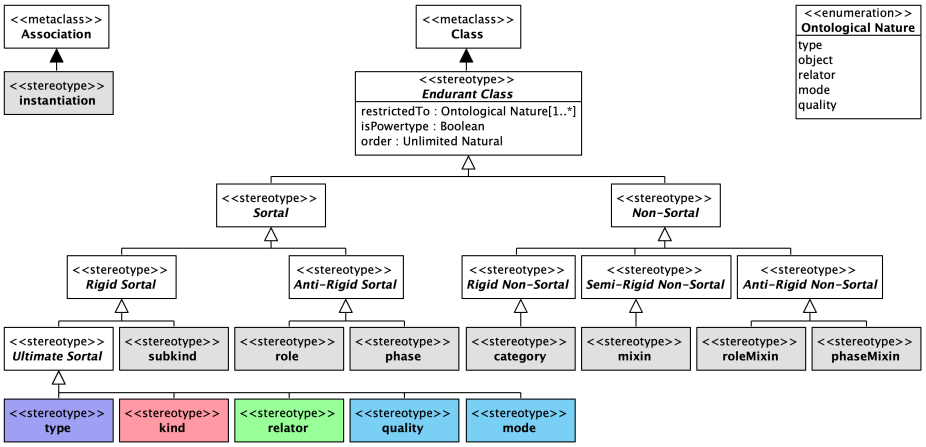

Figure 5: OntoUML profile for high-order domain types.

The attribute fleet of Ship Type corresponds to the number of ships that instantiate the type, and is an example of 'resultant property' [20]. The attributes of Ship Class by Size are examples of the so-called regularity attributes [9, 20], and should be accompanied by OCL constraints relating a ship type's max length, max weight and max draft to a ship's length, weight and draft. These attributes are given values as static features of the instances of Ship Class by Size.

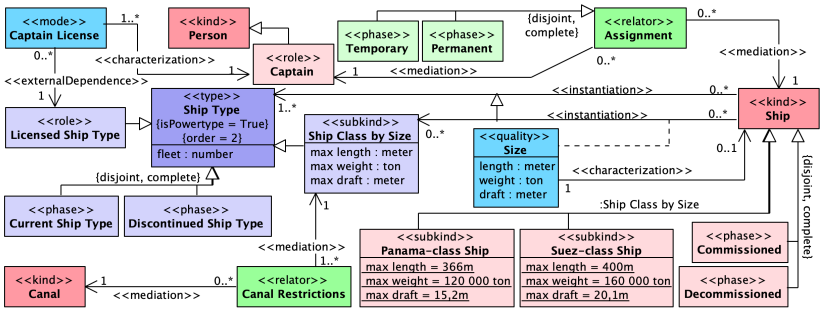

Figure 6: Example including high-order types in the maritime domain.

4.3 Semantically-motivated constraints for high-order types

Here, we discuss a number of consequences of the theory introduced in Section 3 for the various combinations of modeling constructs involving high-order types. These rules can be detected automatically in OntoUML models and rule out unsound models. Some of these rules are direct consequences of the MLT microtheory, some are a direct consequence of considering types as endurants, and some arise from the combination of MLT with UFO constraints applicable to endurant types.

For example, as a direct consequence of MLT alone, a higher-order categorizer always specializes a Cardelli powertype of the same base type [9]. Thus, any type of ship (i.e., any higher-order type related to the base type Ship through «instantiation») will be a specialization of Ship Type, and Ship Type, as a Cardelli powertype, is the most abstract type related to Ship through «instantiation». Further, as a direct consequence of MLT, classes of a different order cannot be related by specialization (so, a specialization of ShipType that also specializes Ship can be ruled out automatically[7]). Likewise, «instantiation» can only relate entities of adjacent orders. See [1] for other rules that can be inferred from MLT, including those involving orderless types. Further rules arising from MLT alone govern the interaction between high-order types and the UML semantics of isCovering and isDisjoint metaproperties of generalization sets [8].

As a straightforward consequence of considering types as endurants is that all constraints applicable to endurant types in OntoUML [22] also apply to high-order types. Hence, anti-rigid (high-order) types cannot specialize rigid (high-order) types, (high-order) non-sortals cannot specialize (high-order) sortals, each (high-order) sortal specializes a unique (high-order) kind, and (high-order) kinds are the most abstract (high-order) sortals in a taxonomy. Further, since every kind is disjoint, classes stereotyped «type», «kind», «relator», «quality» and «mode» cannot specialize each other and must not have common subclasses.

Rules that arise from the combination of MLT with UFO govern the way that previously existing OntoUML stereotypes can be used with the introduced «type» stereotype. For example, consider that a Cardelli powertype imposes a single requirement for its instances: that they specialize the base type, i.e., that they include the intension of the base type in their intension. Since types cannot change their intension while maintaining their identity [9, 20], *any Cardelli powertype must be rigid*, thus classes with isPowerType set to true must be stereotyped «category», «type» or «subkind». (Note that this does not make the *instances* of a Cardelli powertype—e.g., Comissioned, Decomissioned—themselves rigid!)

4.4 Rules involving UFO classes in OntoUML

Including support for high-order types in OntoUML opens up the possibility to represent certain UFO categories themselves in OntoUML. This allow us to settle the ontological categories of instances of high-order types that are not represented explicitly in a model. For example, in the biological taxonomy domain, all instances of Species are kinds, all instances of Breed are subkinds. For specific species or breeds included in the model, stereotypes reveal the applicable ontological categories. However, general statements about species and breeds can be made by Species specializing UFO Kind and Breed specializing UFO Subkind.

In this case, certain rules are applicable for types related by «instantiation», depending on the stereotype of the base type following the analysis in [10]:

– If the *base type is anti-rigid* (i.e., stereotyped «phase», «role», «phaseMixin» or «roleMixin»), the high-order type (whether a Cardelli or an Odell powertype) will

[7] This kind of error may appear simple to prevent, however, there is overwhelming empirical evidence that it occurs very often in practice, see [6, 12].

specialize UFO Anti-Rigid Type[8]; its instances will be stereotyped «phase», «role», «phaseMixin» or «roleMixin» when modeled as the base type's specializations.
- If the *base type is a sortal* (i.e., stereotyped «kind», «relator», «mode», «quality», «kind», «type», «phase» or «role»), the high-order type will specialize UFO Sortal or one of its subcategories *except* for UFO Kind (because the base type already supplies a principle of identity). Its instances, when explicitly modeled as specializations of the base type, will thus either be stereotyped «subkind», «phase» or «role».
- More specifically, *if the base type is a sortal stereotyped «role» or «phase»*, the high-order type will either specialize UFO Phase or Role, and its instances when explicitly modeled will thus either be stereotyped «phase» or «role» accordingly.

Rules can also be formulated in the other 'direction', i.e., rules that apply depending on which UFO type is specialized by the higher-order type that is related to a base type through «instantiation»:

- If *the higher-order type specializes UFO NonSortal*, then its base type must be a non-sortal (i.e., stereotyped «mixin», «phaseMixin» or «roleMixin»).
- If *the higher-order type specializes UFO Rigid Type*, its base type must be rigid.
- More specifically, *if it specializes UFO Kind*, its base type must be a «category» or a semi-rigid «mixin». A powertype that specializes UFO Kind cannot be an overlapping categorizer of a base type, i.e., cannot have an upper bound cardinality higher than one in the association end attached to it in an association stereotype «instantiation». This is because all kinds are pairwise disjoint and thus no overlap in kinds is admissible.

5 Related Work

Multi-level modeling has been an active area of research, with many approaches developed and studied in the last two decades [4, 5, 9, 11, 15, 24, 25, 28], but with roots in the 1990s [3, 14, 30, 33] and 1980s [7]. In this paper, we have incorporated into UFO a key tenet of the multi-level modeling literature: the recognition of entities that are simultaneously types and instances ('clabjects' [3]), leading to the iterated application of instantiation across an arbitrary number of classification levels.

The use of UFO as a foundation allowed us to address the ontological nature of types, an aspect which is not addressed explicitly in the aforementioned approaches, which are neutral as concerns ontological choices (in the sense discussed in [17, 18]). We have proposed an ontological interpretation for types as endurants, which means that the various distinctions for endurant types in UFO also apply to high-order domain types. Note that, none of the related efforts in the multi-level modeling literature incorporates modeling guidelines considering the metaproperties of rigidity and sortality. At the time of writing, OntoUML is the only conceptual modeling language to leverage these metaproperties to multi-level taxonomies. The OntoUML plugin is capable of checking

[8] This class was omitted from Fig. 1, but is a supertype of Anti-Rigid Sortal and Anti-Rigid NonSortal, more specifically, their disjoint union.

the semantically-motivated rules automatically, some of which are direct consequences of MLT, and some of which arise from the combination of MLT with UFO.

Beyond the literature on multi-level modeling, the conceptualization of types of types has also received some attention in the ontology literature. BORO (tracing back to [32]), e.g., is an extensional 4D ontology that uses the concept of *power class* to refer to the class whose instances are all specializations of a class (similar to [7]). Because of the ontological choices of BORO, power classes are not subject to change, which is a key difference to our approach. Types of types are also incorporated into the Cyc ontology [13], including some support for order stratification similar to MLT's which we have incorporated in OntoUML. However, differently from the work reported here, Cyc does not address the various modal aspects of types. Other foundational ontologies, such as BFO [31] and TUpper [16] are defined within traditional two-level schemas and hence do not cater to the representation of domain ontologies with types of types.

The same can be said in general of DOLCE [26], although recent work with this foundational ontology has explored the issue of *concept change* in time with a focus on designed product types [34]. In this work, objects instantiate *concepts* (i.e., domain types) when they manifest the properties collected in that concept [34]. Concepts may change the properties they collect arbitrarily (throughout the design process). In their own words, they "are totally liberal with respect to how concepts can change through time". Because of this, the instantiation relation depends not only on the state of the object being classified but also on the state of the concept. The same can be said of [27] when addressing the notion of *concept drift*. While this intricacy may be required in the domains the authors intended to address, it poses a significant challenge to conceptual modeling as fundamental structural relations like specialization become time-dependent. Here, instead, we consider intensions to be essential (part of the high-order kind), and hence changing a type's intension would amount to the creation of a new type (not identical but historically related to the previous one).

6 Conclusion

This paper proposes an extension of UFO based on a formal multi-level theory dubbed MLT [1, 9]. This extension is motivated by empirical evidence supporting the demand for high-order types in ODCM [6, 12]. Particularly to OntoUML, a recently developed dataset (see Footnote 5) has indicated several ontologies that systematically subvert the language's syntax and rules to capture multi-level features present in subject domains.

The proposed extension revisits the UFO taxonomy by employing MLT's characterization of multi-level schemes in place of UFO's original two-level characterization. Further, we perform an ontological analysis of the concept of Type, proposing an account of *types as endurants*. This analysis incorporates in UFO a preliminary analysis of higher-order types some of us conducted earlier [20], which makes the case for the ontological soundness of this particular interpretation of *types as endurants*, and details the advantages of this interpretation over competing approaches. By formalizing in UFO this interpretation, the ontology is now able to admit types as entities that *exist in time* and that undergo changes without changing their identities. This account also enables the classification of types of types (i.e., high-order types) based on the same modal

features originally recognized in endurant types: (i) sortal high-order types, in contrast to non-sortal types, classify entities conforming to a unique identity principle which it either provides or inherits from another high-order type; (ii) rigid high-order type necessarily classify (in the modal sense) their instances, as opposed to anti-rigid and semi-rigid types. Moreover, (high-order) phases are anti-rigid high-order types based on intrinsic properties of their instances, whilst anti-rigid high-order types based on relational properties are classified as (high-order) roles.

The syntactic constraints we have defined for the profile stem directly from independently developed axiomatizations of UFO [22] and MLT [1, 8, 9]. The result is a rich set of rules that prevent common mistakes in multi-level models and also those involving incorrect combinations of metaproperties. The latter case motivated early ontology-driven approaches such as OntoClean and was originally incorporated in UFO's taxonomy of substantials [19], and recently extended for the taxonomy of endurants [22]. A full formalization of the overall combined theory is a subject of ongoing work.

Acknowledgements

This research is partly funded by Brazilian funding agencies CNPq (313687/2020-0 and 407235/2017-5), CAPES (23038.028816/2016-41) and FAPES (281/2021).

References

1. Almeida, J.P.A., Fonseca, C.M., Carvalho, V.A.: A comprehensive formal theory for multi-level conceptual modeling. In: Conceptual Modeling. ER 2017. vol. 10650. Springer (2017)
2. Almeida, J. P. A. et al.: Events as entities in ontology-driven conceptual modeling. In: 38th Intl. Conf. Conceptual Modeling (ER 2019). LNCS, vol. 11788, pp. 1–16. Springer (2019)
3. Atkinson, C.: Meta-modelling for distributed object environments. In: Proceedings First International Enterprise Distributed Object Computing Workshop. pp. 90–101 (1997)
4. Atkinson, C., Gerbig, R.: Melanie: multi-level modeling and ontology engineering environment. In: Proceedings of the 2nd International Master Class on Model-Driven Engineering: Modeling Wizards. pp. 1–2 (2012)
5. Atkinson, C., Kühne, T.: The essence of multilevel metamodeling. In: International Conference on the Unified Modeling Language. pp. 19–33. Springer (2001)
6. Brasileiro, F., Almeida, J.P.A., Carvalho, V.A., Guizzardi, G.: Applying a multi-level modeling theory to assess taxonomic hierarchies in Wikidata. In: Proc. of the 25th International Conference Companion on World Wide Web. p. 975–980. WWW '16 Companion (2016)
7. Cardelli, L.: Structural subtyping and the notion of power type. In: Proc. 15th ACM SIGPLAN-SIGACT symposium on Principles of programming languages. pp. 70–79 (1988)
8. Carvalho, V.A., Almeida, J.P.A., Guizzardi, G.: Using a Well-Founded Multi-Level Theory to Support the Analysis and Representation of the Powertype Pattern in Conceptual Modeling. In: Proc. 28th Int. CAiSE Conf. LNCS, vol. 9694, pp. 309–324. Springer (2016)
9. Carvalho, V.A., Almeida, J.P.A.: Toward a well-founded theory for multi-level conceptual modeling. Software & Systems Modeling **17**(1), 205–231 (2018)
10. Carvalho, V.A., Almeida, J.P.A., Fonseca, C.M., Guizzardi, G.: Multi-level ontology-based conceptual modeling. Data & Knowledge Engineering **109**, 3–24 (2017)
11. Clark, T., Gonzalez-Perez, C., Henderson-Sellers, B.: A foundation for multi-level modelling. In: Proc. MULTI 2014 co-located with ACM/IEEE MoDELS 2014. CEUR Workshop Proceedings, vol. 1286, pp. 43–52. CEUR-WS.org (2014)

12. Dadalto, A. et al.: Type or individual? evidence of large-scale conceptual disarray in wikidata. In: Conceptual Modeling. pp. 367–377. Springer (2021)
13. Foxvog, D.: Instances of instances modeled via higher-order classes. Foundational Aspects of Ontologies (9-2005), 46–54 (2005)
14. Goldstein, R.C., Storey, V.C.: Materialization. IEEE Trans. Knowl. Data Eng. **6**(5), 835–842 (1994). https://doi.org/10.1109/69.317711
15. Gonzalez-Perez, C., Henderson-Sellers, B.: A powertype-based metamodelling framework. Software and Systems Modeling **5**(1), 72–90 (2006)
16. Grüninger, M., Ru, Y., Thai, J.: TUpper: A top level ontology within standards. Applied Ontology **17**(1), 143–165 (Mar 2022). https://doi.org/10.3233/ao-220263
17. Guarino, N.: The ontological level. In: 16th International Wittgenstein Symposium. pp. 443–456. Hölder-Pichler-Tempsky (1993)
18. Guarino, N.: The ontological level: Revisiting 30 years of knowledge representation. In: Conceptual Modeling: Foundations and Applications - Essays in Honor of John Mylopoulos. vol. 5600, pp. 52–67. Springer (2009)
19. Guizzardi, G.: Ontological foundations for structural conceptual models. TI/CTIT (2005)
20. Guizzardi, G., Almeida, J.P.A., Guarino, N., Carvalho, V.A.: Towards an ontological analysis of powertypes. In: Joint Ontology Workshops (JOWO). vol. 1517. CEUR-WS.org (2015)
21. Guizzardi, G., Wagner, G., Guarino, N., van Sinderen, M.: An ontologically well-founded profile for UML conceptual models. In: Proc. 16th Int. CAiSE Conf. pp. 112–126 (2004)
22. Guizzardi, G.: Endurant types in ontology-driven conceptual modeling: Towards OntoUML 2.0. In: Conceptual Modeling. ER 2018. pp. 136–150. Springer (2018)
23. Guizzardi, G. et al.: Towards ontological foundations for conceptual modeling: the Unified Foundational Ontology (UFO) story. Applied ontology **10**(3-4), 259–271 (2015)
24. Jeusfeld, M.A., Neumayr, B.: Deeptelos: Multi-level modeling with most general instances. In: ER. Lecture Notes in Computer Science, vol. 9974, pp. 198–211 (2016)
25. de Lara, J., Guerra, E.: Deep meta-modelling with metadepth. In: TOOLS (48). Lecture Notes in Computer Science, vol. 6141, pp. 1–20. Springer (2010)
26. Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A.: WonderWeb Deliverable D18. Tech. rep., CNR (2003)
27. Masolo, C., Sanfilippo, E.M., Lamé, M., Pittet, P.: Modeling concept drift for historical research in the digital humanities. In: JOWO. CEUR Workshop Proceedings, vol. 2518. CEUR-WS.org (2019)
28. Neumayr, B., Grün, K., Schrefl, M.: Multi-level domain modeling with m-objects and m-relationships. In: APCCM. CRPIT, vol. 96, pp. 107–116. Australian Computer Society (2009)
29. Object Management Group: Unified Modeling Language (UML) Version 2.5.1. Tech. rep. (Dec 2017), `http://www.omg.org/spec/UML/2.5.1`
30. Odell, J.: Power types. Journal of Object-Oriented Programming **7**(2), 8–12 (1994)
31. Otte, J.N., Beverley, J., Ruttenberg, A.: Bfo: Basic formal ontology. Applied Ontology **17**, 17–43 (2022)
32. Partridge, C.: Business objects: re-engineering for re-use. Butterworth-Heinemann (1996)
33. Pirotte, A., Zimányi, E., Massart, D., Yakusheva, T.: Materialization: A powerful and ubiquitous abstraction pattern. In: 20th International Conference on Very Large Data Bases. p. 630–641. Morgan Kaufmann (1994)
34. Sanfilippo, E.M., Masolo, C., Porello, D.: Design knowledge representation: An ontological perspective. In: Proc. AIDE 2015. vol. 1473. CEUR-WS.org (2015)
35. Schwartz, S.P.: Natural kinds and nominal kinds. Mind **89**(354), 182–195 (1980)
36. Verdonck, M. et al.: Insights on the use and application of ontology and conceptual modeling languages in ontology-driven conceptual modeling. In: Proc. ER. Springer (2016)