Stability Patterns in Ontology-Driven Conceptual Modeling

Giancarlo Guizzardi[1,2] **, João Paulo A. Almeida**[2]

[1] Ontology and Cognitive Modeling Research Group (CORE)
Free University of Bozen-Bolzano
Piazza Domenicani, 3, 39100, Bolzano, Italy

[2]Ontology & Conceptual Modeling Research Group (NEMO)
Federal University of Espírito Santo (UFES)
Av. Fernando Ferrari, 514, 29075-910, Vitória, Brazil

gguizzardi@unibz.it, jpalmeida@ieee.org

Abstract. *Stability is a key quality of a conceptual model. A stable conceptual model is able to withstand changes in domain conceptualization and user requirements without major impact. This paper addresses stability of ontology-driven conceptual models by presenting a number of patterns in the OntoUML language which are derived from characteristics of the foundational ontology underlying the language. The discussed stability patterns include: orthogonal subtype partitions (more specifically phase and subkind partitions), multi-level modeling with high-order types, reification of intrinsic and relational aspects, and model taxonomy refactoring with non-sortal types.*

1. Introduction

Significant effort is invested in the production and maintenance of reference conceptual models. Because of this, a key quality of a conceptual model is its stability, i.e., its ability to withstand change without major disruption. Change to a conceptual model is often required as a result of evolution in the conceptualization of the model's subject domain or as a natural revision of user requirements imposed on the model. Model stability is particularly critical when the model is used as a blueprint for system integration and implementation, and changes to the model affect not only maintenance operations on the model itself, but also can lead to costly implementation changes.

This paper discusses stability patterns in ontology-driven conceptual models. In particular, we present and discuss a number of ontology-driven stability patterns in the OntoUML ontology-driven conceptual modeling language. These patterns are motivated by their potential in fostering the stability of models and are justified based on their underlying ontological semantics, derived from the ontological categories in the Unified Foundational Ontology (UFO) [Guizzardi 2005b, Guizzardi et al. 2015b].

This paper is further structured as follows. Section 2 provides a brief introduction to the OntoUML language, introducing the ontological categories which are used to articulate the patterns presented in Section 3. The patterns are presented with a number of examples from different subject domains, and include: orthogonal subtype partitions (more specifically phase and subkind partitions), multi-level modeling with high-order types, reification of intrinsic and relational aspects, and taxonomies with non-sortal types. Section 4 discusses some related work. Finally, Section 5 presents some concluding remarks.

2. A Brief Introduction to OntoUML

OntoUML is a language whose meta-model has been designed to comply with the ontological distinctions and axiomatization of a theoretically well-grounded foundational ontology named UFO (Unified Foundational Ontology) [Guizzardi 2005b, Guizzardi et al. 2015b]. UFO is an axiomatic formal theory based on contributions from Formal Ontology in Philosophy, Philosophical Logics, Cognitive Psychology, and Linguistics. OntoUML has been successfully employed in several industrial projects in different domains, such as petroleum and gas, digital journalism, complex digital media management, off-shore software engineering, telecommunications, retail product recommendation, and government [Guizzardi et al. 2015b]. A recent study shows that UFO is the second-most used foundational ontology in conceptual modeling and the one with the fastest adoption rate [Verdonck and Gailly 2016]. That study also shows that OntoUML is among the most used languages in ontology-driven conceptual modeling.

In the sequel, we briefly explain a selected subset of the ontological distinctions put forth by the Unified Foundational Ontology (UFO). We also show how these distinctions are represented by the modeling primitives of OntoUML (as a UML profile). For an in-depth discussion, philosophical justifications, formal characterization and empirical support for these categories one should refer to [Guizzardi 2005b, Guarino and Guizzardi 2015].

Take a domain in reality restricted to endurants [Guizzardi 2005b] (as opposed to events or occurrents). Central to this domain we will have a number of object *Kinds*, i.e., the genuine fundamental types of objects that exist in this domain. The term "kind" is meant here in a strong technical sense, i.e., by a kind, we mean a type capturing essential properties of the things it classifies. In other words, the objects classified by that kind could not possibly exist without being of that specific kind.

Kinds tessellate the possible space of objects in that domain, i.e., all objects belong to exactly one kind and do so necessarily. Typical examples of kinds include 'Person', 'Organization', and 'Car'. We can, however, have other static subdivisions (or subtypes) of a kind. These are naturally termed *Subkinds*. As an example, the kind 'Person' can be specialized in the subkinds 'Man' and 'Woman' (Figure 1). Disjoint and Complete generalization sets comprising subkinds as the specializing classes are called *subkind partitions*.

Object kinds and subkinds represent essential properties of objects (they are also termed rigid or static types [Guizzardi 2005b]). We have, however, types that represent contingent or accidental properties of objects (termed anti-rigid types [Guizzardi 2005b]). These include *Phases* (for example, in the way that 'being a living person' captures a cluster of contingent *intrinsic* properties of a person, or in the way that 'being a puppy' captures a cluster of contingent *intrinsic* properties of a dog) and *Roles* (for example, in the way that 'being a husband' captures a cluster of contingent *relational* properties of a man participating in a marriage). In other words, the difference between the contingent properties represented by a phase and a role is the following: phases represent properties that are intrinsic to entities (e.g., 'being a puppy' is being a dog that is in a particular developmental phase; 'being a living person' is being a person who has the intrinsic property of being alive; 'being available car' is being a car that is functional and, hence, can be rented); roles, in contrast, represent properties that entities have in a relational context, i.e., contingent relational properties (e.g., 'being a husband' is to bear a number of

commitments and claims towards a spouse in the scope of a marital relationship; 'being a student' is to bear a number of properties in the scope of an enrollment relationship with an educational institution).

Kinds, Subkinds, Phases, and Roles are categories of object *Sortals*. In the philosophical literature, a sortal is a type that provides a uniform principle of identity, persistence, and individuation for its instances [Guizzardi 2005b]. To put it simply, a sortal is either a kind (e.g., 'Person') or a specialization of a kind (e.g., 'Student', 'Teenager', 'Woman'), i.e., it is either a type representing the essence of what things are or a subclassification applied to the entities that "have that same type of essence".

Objects can relate to each other via parthood relations forming partonomic structure (e.g., a car can be composed of chassis and engine). These parthood relations can be optional (in both directions), mandatory (in both directions), as well as implying existential dependence (in both directions). For example, a car has spare tyres and navigational systems as optional parts, engine as mandatory parts (every car must have an engine but that does not have to always be same engine), and chassis as an *essential* part (since legally the identity of the car is tied to the identity of the chassis, every car must have *that specific* chassis as part) [Guizzardi 2005b].

Relators (or relationships in a particular technical sense [Guarino and Guizzardi 2015]) represent clusters of relational properties that "hang together" by a nexus (provided by a relator kind). Moreover, relators (e.g., marriages, enrollments, presidential mandates, citizenships, are full-fledged *Endurants*. In other words, entities that endure in time bearing their own essential and accidental properties and, hence, first-class entities that can change in a qualitative manner while maintaining their identity.

As discussed in depth in [Guarino and Guizzardi 2015], relators are the truthmakers of relational propositions, and relations (as classes of n-tuples) can be completely derived from relators [Guizzardi 2005b]. For instance, it is 'the marriage' (as a complex relator composed of mutual commitments and claims) between 'John' and 'Mary' that makes true the proposition that "John is the husband of Mary". Relators are existentially dependent entities (e.g., the marriage between John and Mary can only exist if John and Mary exist) that bind together entities (their relata) by the so-called *mediation* relations—a particular type of existential dependence relation [Guizzardi 2005b]. As discussed in depth in [Guarino and Guizzardi 2015], all domain relations in business models (the so-called material relations) can be represented exclusively by employing relators and these existential dependence relations (mediation).

Objects participate in relationships (relators) playing certain "roles". For instance, people play the role of spouse in a marriage relationship; a person plays the role of president in a presidential mandate. 'Spouse' and 'President' (but also typically student, teacher, pet) are examples of what we technically term a role in UFO, i.e., a relational contingent sortal (since these roles can only be played by entities of a unique given kind). There are, however, relational and contingent role-like types that can be played by entities of multiple kinds. An example is the role 'Customer' (which can be played by both people and organizations). We call these role-like types that classify entities of multiple kinds *Role Mixins*. In general, types that can possibly classify entities of multiple kinds are

termed non-sortals. Besides Role Mixins (which are, again, anti-rigid and relational dependent), OntoUML countenances: (i) anti-rigid and relationally independent non-sortals termed *Phase Mixins* (e.g., Functional and Non-Functional device as types contingely classifying devices of different kinds) [Guizzardi et al. 2018]; (ii) rigid non-sortals termed *Categories* (e.g., being physical object is necessary for all physical objects but it classifies entities of multiple kinds, e.g., cars, people, buildings, houses); (iii) semi-rigid non-sortals termed *Mixins*. Semi-rigidity means that the type classifies some of its instances necessarily while classifying others contingently. An example is *Legally Recognized Conjugal Relationship*, which is essential for Marriages but contingent for Civil Partnerships (only long-term civil partnerships are legally recognized) [Guizzardi et al. 2018].

Relators are examples of *Aspects* (moments [Guizzardi 2005b], variable tropes [Guarino and Guizzardi 2015]), i.e., endurants that are existentially dependent on other endurants. Relators are, in fact, existentially dependent on a multitude of other endurants and that is why can play the aforementioned role of truth-makers of relations. In other words, relators are *relational aspects*. There are, however, aspects that are intrinsic, i.e., existentially dependent on a single object which is then termed their bearer. An intrinsic aspect is tied to its bearer via the existential dependence relation of *inherence/characterization*. Instrinsic aspects can be further specialized into *qualities* and *modes*. Qualities (e.g., colors, weights, electric charge) can be thought of as reified qualitative aspects of their bearers that can change in time taking different *(quality) values* in certain quality spaces (e.g., the color spindle, the taste tetrahedron, a structure isomorphic to a subset of positive half-line of real numbers for weight). Modes are object-like entities that frequently bear their own aspects, but which are existentially dependent on other endurants (e.g., my headache, Sofia's ability to play the guitar, Matteo's intention to play Fortnite tonight, the disposition of a magnet to attract metallic material).

Finally, UFO (and, hence, also OntoUML) countenances the existence of types whose instances are other types (i.e., higher-order types) forming multi-level structures. For example, the type 'Bird Species' has as instances not individual birds but types whose instances are individual birds such as 'Emperor Penguin' and 'American Eagle' [Guizzardi 2005b]. A particular bird (e.g., my pet eagle) is connected to an instance of the higher-order type 'Bird Species' by an *instantiation* relation [Carvalho et al. 2016]. In line with original UFO terminology [Guizzardi and Wagner 2004], we use in this paper the stereotype *classificationType* to represent these higher-order types.

3. Stability Patterns

3.1. Orthogonal Generalization sets

Subtyping in OntoUML is a logical relation and, hence, an instance can instantiate multiple types (but not multiple kinds). Moreover, we can easily generate generalization sets that partition a common supertype using orthogonal specialization criteria. Two common ways of doing that are by using the *phase pattern* [Ruy et al. 2017] (a phase partition, i.e., a dynamic partition—with a common supertype) and the *subkind pattern* [Ruy et al. 2017] (a subkind partition, i.e., a static partition—with a common supertype). An examples instantiated both of these patterns is illustrated in figure 1 below.

As one can observe, this model easily avoids the proliferation of *intersection classes* [Olivé 2007]. These are classes that follow the pattern of *derivation by inter-*

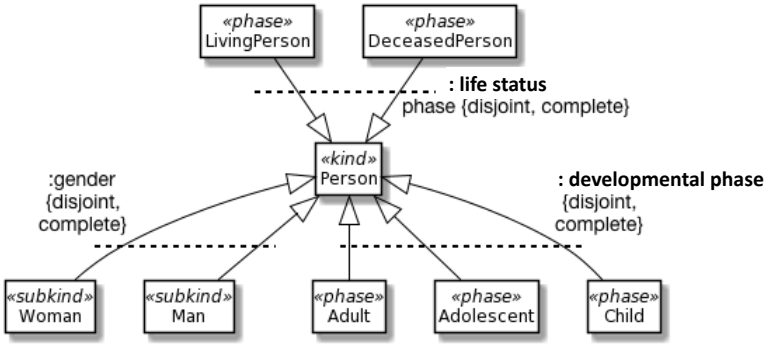

Figure 1. Orthogonal Generalization sets*

section, and which abounds in taxonomic structures in languages that do not allow for multiple classification. Examples in an alternative model of this very same domain would be *Adult Living Man*, *Adolescent Living Man*, *Adult Deceased Woman*, and so on. Besides the proliferation of intersection types, we have also some arbitrariness in the construction of these taxonomic structures. For example, how to justify first specializing, for example, *Man* into *Living Man* and *Deceased Man* and then *Living Man* into *Adult Living Man* as opposed to specializing *Man* into *Male Child*, *Male Adolescent* and *Male Adult* and then, for example, *Male Adult* into *Living Male Adult* and *Deceased Male Adult*[1]? Finally, in addition to their proliferation and arbitrariness, a change in any of these partitions (e.g., the introduction of a new development phase) would cause multiple consequent changes (one per each possible new intersection type, combinatorially).

3.2. Multi-Level Modeling

Alternatively, of course, the model above can be represented by reducing subtypes to members of an enumeration [Halpin 2007]. A better way to do that is to use the multi-level support existent in OntoUML, in line with what has been proposed in [Carvalho et al. 2016, Guizzardi et al. 2015a] (see Figure 2). In this example, the subtypes of 'Person' in the corresponding generalization sets become instances of these higher-order types. The disadvantage here is the impossibility in the model itself (except by the use of constraints) to represent eventual attributes and relations involving these subtypes. This problem is addressed by using or model reification approach as discussed in the next section.

3.3. Reification

OntoUML has a full support for the reification of properties, both intrinsic and relational. On the side of intrinsic properties, we have a distinction between *qualities* and *modes*.

The former allows for the separation between a quality (e.g., the color of an object as itself represented as a first-class entity) and the value of that quality into appropriate conceptual spaces (see Figure 3). The pattern represented in Figure 3 (and which was first proposed in [Guizzardi et al. 2006]), allows for isolating an attribute represented

[1]Notice that the decision of making the first partition with *Man* and *Woman* is taken for granted in this case, given that anti-rigid types (e.g., phases) cannot be supertypes of rigid types (e.g., subkinds) [Guizzardi 2005b].

***Erratum: the labels "development phase" and "life status" were swapped by mistake in the original figure.**

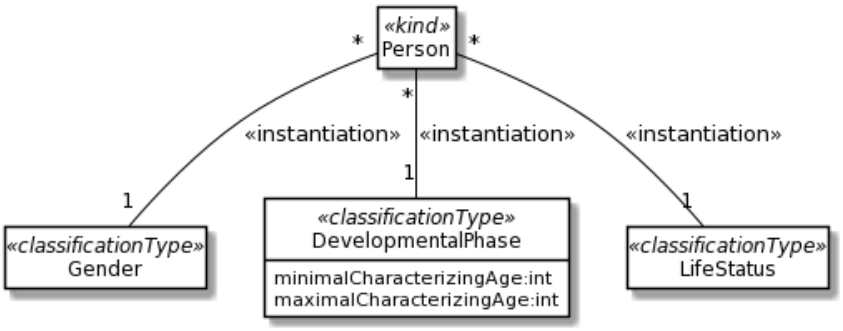

Figure 2. Representing Phase and Subkind partitions using the *Powertype pattern* in OntoUML

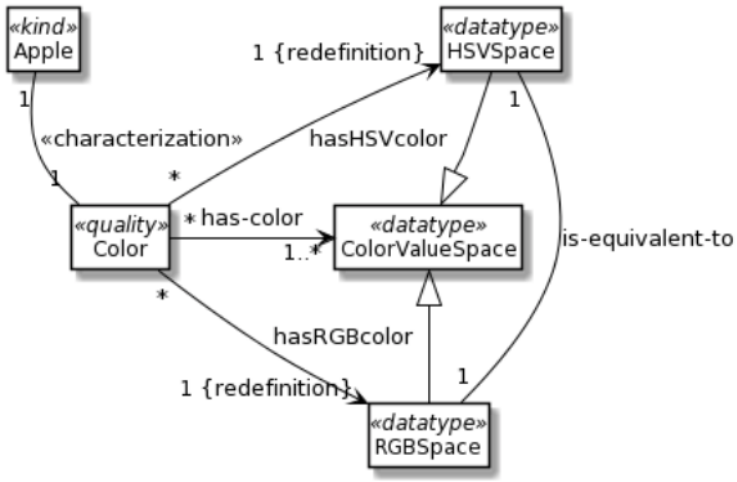

Figure 3. Qualities and Quality Spaces

by a quality from the multiple spaces of values (e.g., RGB, HSV) on which that attribute/quality can be projected. As a consequence, changes in quality spaces do not propagate to the quality at hand. In contrast, imagine a model in which we have *n* colored object types, all of which have an attribute 'color' taking value in a RGB space. Now, suppose we want to replace this attribute value space of color to HSV. This one change would propagate to these *n* types triggering changes in all manifestations of this attribute[2].

Moreover, associated with dynamic types (*roles* and *phases*), we have particular types of *modes*. For example, John plays the role of Husband because there is a marriage *relator* binding him to his wife. His wife, Mary, plays that role because of this very same relator [Guarino and Guizzardi 2015]. As discussed in depth in [Guizzardi 2005a], this marriage, on its turn, is a sum of two modes: *John-qua-husband-of-Mary* and *Mary-qua-wife-of-John*. Each of these modes aggregates all the properties (e.g., rights, obligations) that they both have in the scope of that relation. In an analogous manner, other types of modes can be related to phase-instantiation (e.g., it is the presence and/or the symptomatic state of a disease inhering in me that makes me instantiate the phase 'Ill Person', as opposed to 'Healthy Person').

[2]An alternative would be to have a common supertype of all these types termed 'Colored Object' as a single place for this 'color' attribute. This solution is discussed in Section 3.4 below.

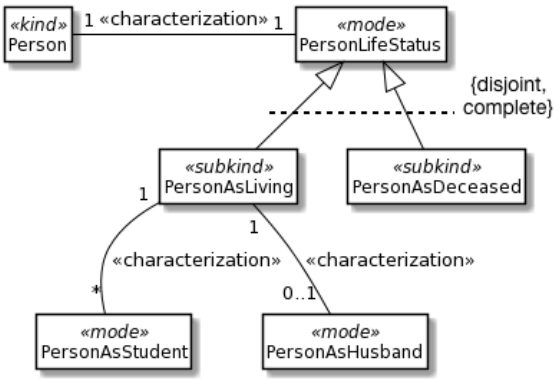

Figure 4. Mode Reification

The reification of modes provides for yet another mechanism managing taxonomic structures (see Figure 4). Such a model allows for separating the properties an individual has *essentially* (associated to that individual's kind) from the properties it instantiates contingently in the scope of roles and phases. Moreover, it allows for isolating these accidental properties in each of these contexts (e.g., the properties John has as husband, as student as living person, etc)[3].

OntoUML also allows for the reification of relationships as relators. In fact, it requires the reification of relationship of a particular type of relations termed *material relations*, which are by far the most common relation in application domains [Guarino and Guizzardi 2015].

Relators are the *truthmakers* of material relations [Guarino and Guizzardi 2015]. As discussed in depth in [Guizzardi 2005b], their explicit representation allows for addressing a number of problems that are present in all conceptual modeling representations, including ambiguity in cardinality constraints, and delimiting the scope of transitivity of parthood. But, in addition, it allows for the representation of the so-called *anadyc relations* in a explicit manner.

Anadyc relations are relations whose arity (e.g., binary, ternary) varies from instance to instance of the relation. Take, for example, the case, depicted in Figure 5. Suppose we want to represent that a 'Musician' can play with several musicians. Now, at each instance of this relation, we could have two people playing together (e.g., John, Paul), three people playing together (e.g., John, Paul, George), four people playing together (e.g., John, Paul, George, Ringo). Notice that the (incomplete) model of Figure 5 cannot capture that. In order to do that, we would need instead to either: (a) define a bi-

[3]The patterns proposed here are often used in combination. For example, suppose that we have a model in which the kind *Person* is specialized into the role *Worker*, which, in turn, is specialized, in one dimension, w.r.t. her *type of work contract* (e.g., employment vs. freelance work contract), and, on another dimension, w.r.t. her *professional capacity* (e.g., programmer, business analyst). In this case, we can use this reification pattern to produce the mode type *Person-qua-Worker*, which is further specialized into the orthogonal generalizations sets *type of work contract* (having *Person-qua-Employee* and *Person-qua-Freelance Worker* as subtypes) and *professional capacity* (having *Person-qua-Programmer* and *Person-qua-Business Analyst* as subtypes). One should notice, however, that, since roles are relationally dependent types, in a realistic model of these domains, the *qua individuals* [Guizzardi 2005a] associated to these roles should be modeled as part of the relators associated with the latter.

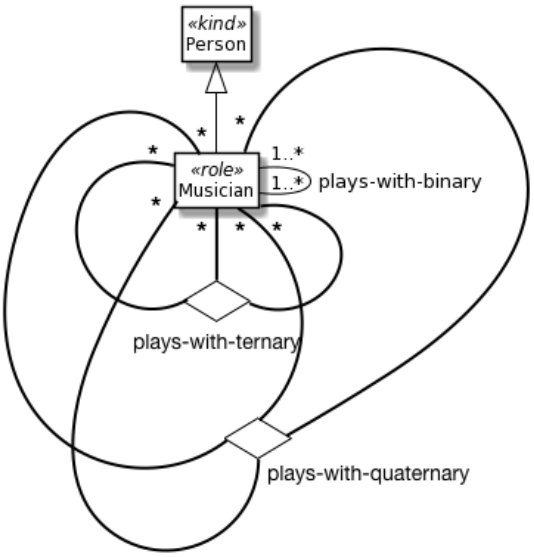

Figure 5. The limitations of traditional approaches in representing anadyc relations in traditional notations

nary relation (e.g., 'plays with', in the model) that captures, for example, that John plays with Paul, Paul plays with John, John plays with George, etc. However, this model would not capture that they are playing together in the very same instance of the relation (i.e., the same musical session); or, (b) we would have to define a large number of relations for each of the possible arities (i.e., plays-binary, plays-ternary, etc.), and, in the case of constraints involving this anadyc relation, adapt these contraints for each of these arity-specific manifestations of that relation. Still, if we find out later the existence of instances of that relation with an unforeseen arity, we would need to create change the model, by including a new relation, with its cardinality constraints, etc.

Given the aforementioned guidelines of the OntoUML language regarding the explicit representation of material relations, this problem simply disappears.This is because we would explicitly reify the relationship itself (see Figure 6) and all the information regarding that relation could be derived from it.

Figure 6. Representing Anadyc Relations with Relators.

3.4. Non-Sortals

In OntoUML, non-sortals are types that represent properties shared by entities of different kinds. Since they cannot provide a uniform principle of identity for all their instances

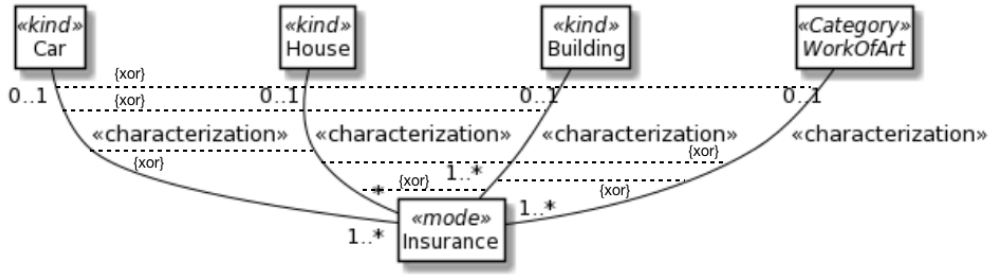

Figure 7. A model in which things of different types can be insured.

(which are provided by each of the respective kinds), non-sortals are abstract types. Standard UML, although having the notion of abstract class, has strong constraints such as "every abstract class must be specialized by a concrete class". This prevents us from creating elegant and stable structures such as the ones discussed in the sequel using non-sortals.

Suppose the situation depicted in the model of Figure 7. In this model, we have that Cars, Houses, Buildings and Works of Art can be insured. An insurance instance (a particular relator) is necessarily connected to an instance of one of the classes (due to existential dependence). However, it is not connected at all to instances of the other $n-1$ classes. In other words, a particular insurance is either a Car insurance, a House insurance, etc. However, by being a car insurance, for example, it is neither connected to a house, nor a building, nor a work of art. As a consequence, the minimum cardinality constraint of the association end that supposedly represent the entity types of the entities mediated by this relator must necessarily be zero. However, since relators are existentially dependent entities, it must be one for at least one of them. A solution to this problem is to use $\{xor\}$ constraints in UML. The problem, in this case, is that, in UML, these constraints are defined between pairs of associations (they are binary constraints). This means that we have to define *combination(n,2)* $\{xor\}$ constraints (where n is the number of kinds of possibly insurable things). Now, if we find out that a new type of entity can be insured (e.g., body parts), we would have to include a new relation between that and 'Insurance' and $n-1$ new $\{xor\}$ constraints between that relation and all the others[4].

In contrast, if we model this situation using non-sortals, we arrive at the model of Figure 8. In this case, if we want to capture that situation that now also body parts can be insured, this only requires: (i) making 'Body Part' as a subtype of the non-sortal 'Insurable Item'—in case body parts are necessarily insured, or (ii) creating a dynamic (anti-rigid) subtype of 'Body Part' and make that a subtype of 'Insurable Item'—in case body parts are only contingently insured (see the analogous example for 'House' in Fig. 8).

As another example, take the model of Figure 9. This model can be easily refactored as the model of Figure 10 with full support for non-sortals. Again, in this example, we can easily include new organs in the model without multiplying the number of new

[4]An alternative to this model would be to specialize 'Insurance' in n-subtypes of 'Insurance' (e.g., 'Car Insurance', 'House Insurance', etc.). Still in this case, the introduction of a new kind of insurable thing (e.g., body parts) would require the creation of a new insurance subtype and the corresponding relations and constraints.

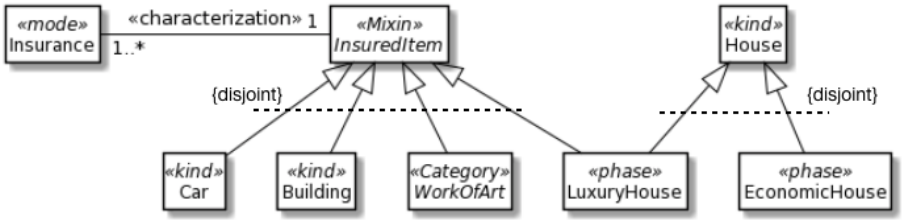

Figure 8. A model in which things of different types can be insured represented using Non-Sortals.

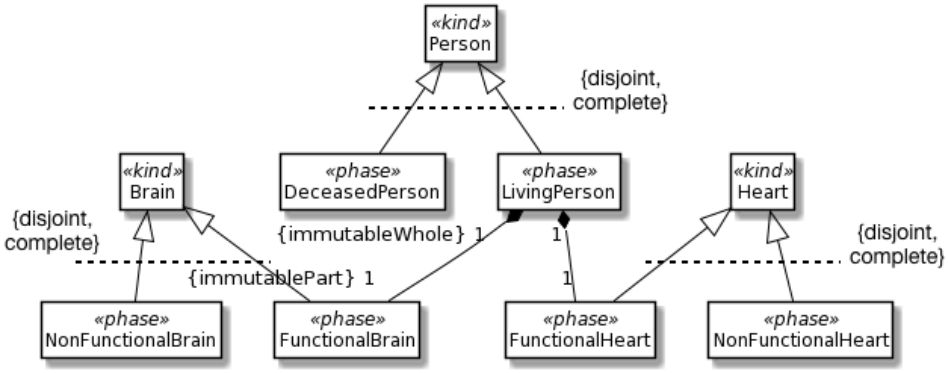

Figure 9. A model representing the relation between people and their functional organs.

classes and relations that need to be introduced because of those changes.

4. Related Work

The work proposed in this paper is in line with the *Normalized Systems Theory (NST)* proposed in [Mannaert et al. 2012, De Bruyn et al. 2018]. This theory is built around four basic axioms, namely: (i) *Separation of Concerns*; (ii) *Data version Transparency*; (iii) *Action version Transparency*; (iv) *Separation of States*.

In [De Bruyn et al. 2018], NST is applied to structural (data) models. In that scope:

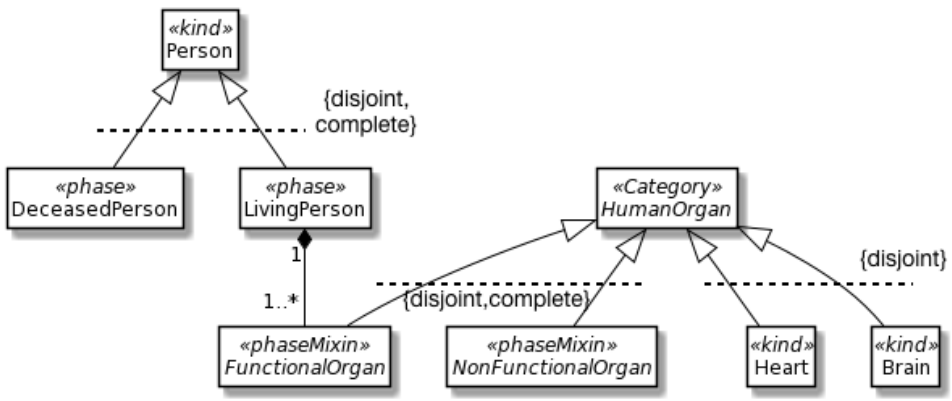

Figure 10. A refactoring of the model of Figure 9 using non-sortals.

- (i) recommends the isolation of *change drivers*, or aspects that can independently change. Separation of concerns is directly reflected, for example, in our use of ortoghonal generalization sets;

- Regarding (ii), the authors recommends the separation, for example, of core entities and (what they term) taxonomic entities. In our proposal, this is achieve directly, for example, in the pattern of figure 2, in which these so-called 'taxonomic entities' are represented as classification types;

- Regarding (iii), "the Version Transparency theorems require that the dependencies between the (modeling) constructs are made explicit" [De Bruyn et al. 2018]. In our approach, this is implied by the very ontological semantics of (intrinsic and relational) aspects in UFO: both *characterization* (connecting qualities and modes to their bearer) as well as *mediation* (connecting relators to their mediated entities) are examples of *existential dependence* relations. This is also the case of the rich set of part-whole relations in which these modal notions are explicitly represented, for example, differentiating mandatory parthood (generic dependence) and essential/inseparable parthood (existential dependence) [Guizzardi 2005b]. Aspects also allows for a form of *Separation of States*, given that phase-inducing modes and qua-entities isolate forms of *aspectual slices* of a core entities (their bearer). For instance, the mode 'personAsStudent' in figure 4 represents in an isolated fashion all the properties that a given (living) person bears in the scope of given *studentship*. State changes there are not propagated even to other studentships of the same person, let alone to other independent modes. In fact, a variant of this reification approach has been applied in [Guizzardi and Zamborlini 2014] to exactly contain change propagation;

- Finally, another aspect put forth here but not explicitly addressed in [De Bruyn et al. 2018] is the role of model refactoring enabled by the use of non-sortals in taxonomic structures. As demonstrated here, these types allow for refactoring operations that isolate sources of change in non-sortal types. Since non-sortals represent properties that are shared by entities of multiple kinds, but using then we naturally move these shared properties from these multiple kinds to a single point of change attached to this non-sortal.

5. Final Considerations

This paper presented a number of patterns that foster the stability of ontology-driven conceptual models. Finding the correct information modeling structures is fundamental for conceptual modeling, in particular, given their role as the corner stone for high-quality information systems [Olivé 2007]. The opposite of stability (or *flexibility* - as easy incorporation of changes [Moody 2003]) is having systems that are prone to what is sometimes termed *change amplification* [Ousterhout 2018] or the so-called *ripple effects* [De Bruyn et al. 2018] (described as a characteristic of artifacts in which even a simple change can require multiple subsequent changes to many components).

In this paper, by adopting a rich ontological semantics for the conceptual modeling language, patterns can be articulated exploring a number of distinctions in the ontological

categories supported in the language through stereotypes. The patterns result in models that can withstand change in a number of scenarios. Orthogonal *subkind* and *phase partitions*, for example, lead to independent taxonomies such that a change in one partition does not affect other partitions. High-order types in multi-level models allows a modeler to focus on invariant aspects of these types, instead of fixing them in a model. When change is expected for these types (instances of the high-order types in the model), the resulting multi-level model is more stable, and the introduction of new types is accommodated in a straightforward manner (not unlike object instantiation). Reification of intrinsic aspects can produce models that resist change in quality spaces employed to conceptualize quality measurement. Finally, reification of relations can solve a particular acute problem with stability of models in the presence of anadyc relations.

References

Carvalho, V. A., Almeida, J. P. A., and Guizzardi, G. (2016). Using a well-founded multi-level theory to support the analysis and representation of the powertype pattern in conceptual modeling. In *International Conference on Advanced Information Systems Engineering*, pages 309–324. Springer.

De Bruyn, P., Mannaert, H., Verelst, J., and Huysmans, P. (2018). Enabling normalized systems in practice–exploring a modeling approach. *Business & Information Systems Engineering*, 60(1):55–67.

Guarino, N. and Guizzardi, G. (2015). "we need to discuss the relationship": Revisiting relationships as modeling constructs. In *International Conference on Advanced Information Systems Engineering*, pages 279–294. Springer.

Guizzardi, G. (2005a). Agent roles, qua individuals and the counting problem. In *International Workshop on Software Engineering for Large-Scale Multi-agent Systems*, pages 143–160. Springer.

Guizzardi, G. (2005b). Ontological foundations for structural conceptual models.

Guizzardi, G., Almeida, J. P. A., Guarino, N., and de Carvalho, V. A. (2015a). Towards an ontological analysis of powertypes. In *JOWO@ IJCAI*.

Guizzardi, G., Fonseca, C. M., Benevides, A. B., Almeida, J. P. A., Porello, D., and Sales, T. P. (2018). Endurant types in ontology-driven conceptual modeling: Towards ontouml 2.0. In *International Conference on Conceptual Modeling*, pages 136–150. Springer.

Guizzardi, G., Masolo, C., and Borgo, S. (2006). In defense of a trope-based ontology for conceptual modeling: an example with the foundations of attributes, weak entities and datatypes. In *International Conference on Conceptual Modeling*, pages 112–125. Springer.

Guizzardi, G. and Wagner, G. (2004). A unified foundational ontology and some applications of it in business modeling. In *CAiSE Workshops (3)*, pages 129–143.

Guizzardi, G., Wagner, G., Almeida, J. P. A., and Guizzardi, R. S. (2015b). Towards ontological foundations for conceptual modeling: The unified foundational ontology (ufo) story. *Applied ontology*, 10(3-4):259–271.

Guizzardi, G. and Zamborlini, V. (2014). Using a trope-based foundational ontology for bridging different areas of concern in ontology-driven conceptual modeling. *Science of Computer Programming*, 96:417–443.

Halpin, T. (2007). Subtyping revisited. In *Proc. CAiSE*, pages 131–141.

Mannaert, H., Verelst, J., and Ven, K. (2012). Towards evolvable software architectures based on systems theoretic stability. *Software: Practice and Experience*, 42(1):89–116.

Moody, D. L. (2003). The method evaluation model: a theoretical model for validating information systems design methods. In *ECIS*.

Olivé, A. (2007). *Conceptual modeling of information systems*. Springer Science & Business Media.

Ousterhout, J. (2018). *A Philosophy of Software Design*. Yaknyam Press.

Ruy, F. B., Guizzardi, G., Falbo, R. A., Reginato, C. C., and Santos, V. A. (2017). From reference ontologies to ontology patterns and back. *Data & Knowledge Engineering*, 109:41–69.

Verdonck, M. and Gailly, F. (2016). Insights on the use and application of ontology and conceptual modeling languages in ontology-driven conceptual modeling. In *International Conference on Conceptual Modeling*, pages 83–97. Springer.