Ontological Patterns, Anti-Patterns and Pattern Languages for Next-Generation Conceptual Modeling

Giancarlo Guizzardi
Ontology and Conceptual Modeling Research Group (NEMO), Computer Science Department,
Federal University of Espírito Santo (UFES), Vitória - ES, Brazil
gguizzardi@inf.ufes.br

Abstract. This paper addresses the complexity of conceptual modeling in a scenario in which semantic interoperability requirements are increasingly present. It elaborates on the need for developing sound ontological foundations for conceptual modeling but also for developing complexity management tools derived from these foundations. In particular, the paper discusses three of these tools, namely, ontological patterns, ontological anti-patterns and pattern languages.

Keywords: Conceptual Modeling, Formal Ontology, Patterns, Anti-Patterns and Pattern Languages

"To begin on a philosophical plane, let us note that we usually behave as if there were three realms of interest in data processing: the real world itself, ideas about it existing in the minds of men, and symbols on paper or some other storage medium. The latter realms are, in some sense, held to be models of the former. Thus, we might say that data are fragments of a theory of the real world, and data processing juggles representations of these fragments of theory…The issue is ontology, or the question of what exists."(G.H. Mealy, Another Look at Data, 1967) [1].

1. Introduction

Information is the foundation of all rational decision-making. Without the proper information, individuals, organizations, communities and governments can neither systematically take optimal decisions nor understand the full effect of their actions. In the past decades, information technology has played a fundamental role in automating an increasing number of information spaces. Furthermore, in the past decades, there has been a substantial improvement in information access. This was caused not only by the advances in communication technology but also, more recently, by the demands on transparency and public access to information.

Despite these advances, most of these automated spaces remained as independent components in large and increasingly complex silo-based architectures. The problem with this is that several of the critical questions we have nowadays in large corporations, government and even professional communities (e.g., scientific communities) can only be answered by precisely connecting pieces of information distributed over these silos. Take for example the following question: from all the outsourcing contracts signed by a government organizational unit with private parties, which ones include parties that made a donation to the political campaign of any individual with power of

decision over that contract? The information needed to answer this question typically exists *"in the ether"*, i.e., in the set of information represented by an existing set of information systems. Moreover, given the current requirements for data transparency, this information is typically even public. However, it usually only exists in dispersed form in a number of autonomous information silos. As consequence, despite the increasing amount of information produced and acquired by the entities, as well as the improvements in information access, answering critical questions such as this one is still extremely hard. In practice, they are still answered in a case-by-case fashion and still require a significant amount of human effort, which is slow, costly and error-prone. The problem of combining independently conceived information spaces and providing unified analytics over them is termed the problem of *Semantic Interoperability*. As reflected in OMG's SIMF RFP [2]: *"the overall human and financial cost to society from our failure to share and reuse information is many times the cost of the systems' operation and maintenance"*.

I use the term Information System here in a broader sense that includes also *Socio-technical Systems*. Moreover, I subscribe here to the so-called *representation view* of information systems [3]. Following this view, an information system is a representation of a certain *conceptualization* of reality. To be more precise, an information system contains information structures that represent *abstractions* over certain portions of reality, capturing aspects that are relevant for a class of problems at hand. There are two direct consequences of this view. Firstly, the quality of an information system directly depends on how truthful are its information structures to the aspects of reality it purports to represent. Secondly, in order to connect two information systems A and B, we first need to understand the precise relation between the abstractions of entities in reality represented in A and B. For instance, suppose A and B are two different systems recording city indicators for two different cities, and that we have to compare the student/teacher ratios in these two cities. In order to do that, we must understand what is the relation between the terms *Student* and *Teacher* as represented in A versus these two terms as represented in B. Understanding this relation requires precisely understanding the relation between the referents in a certain conceptualization of reality represented by these terms. Even a simple indicator such as this one can hide a number of subtle meaning distinctions as explained in [4]: *"One problem is whether "student" refers to full time students, or part time students…it is also difficult to compare an indicator for a single city across time if the definition of student changes. For example, today the educational system includes students with special needs, but 60 years ago they may not have been enrolled."*

In his ACM Turing Award Lecture entitled *"The Humble Programmer"* [5], E. W. Dijkstra discusses the sheer complexity one has to deal with when programming large computer systems. His article represented an open call for an acknowledgement of the complexity at hand and for the need of more sophisticated techniques to master this complexity. Dijkstra's advice is timely and even more insightful in our current scenario, in which semantic interoperability becomes a pervasive force driving and constraining the process of creating information systems in increasingly complex combinations of domains. More and more, information systems are created either by combining existing autonomously developed subsystem, or are created to eventually serve as

components in multiple larger yet-to-be-conceived systems. In this scenario, information systems engineering, in particular, and rational governance, in general, cannot succeed without the support of a particular type of discipline. A discipline devoted to establish well-founded theories, principles, as well as methodological and computational tools for supporting us in the tasks of understanding, elaborating and precisely representing the nature of conceptualizations of reality, as well as in tasks of negotiating and safely establishing the correct relations between different conceptualizations of reality. On one hand, this discipline should help us in producing representations of these conceptualizations that are *ontologically consistent*, i.e., that represent a worldview that aggregates a number of abstractions that are consistent with each other. On the other hand, it should help to make explicit our *ontological commitments*, i.e., to make explicit what exactly is the worldview to which we are committing. In summary, this discipline should help to produce concrete representation artifacts (models) of conceptualizations of reality that achieve the goals of *intra-worldview consistency* and *inter-worldview interoperability*.

The discipline to address the aforementioned challenges is the discipline of *Conceptual Modeling*. However, in order to do that, conceptual modeling languages, methodologies and tools must be informed by another discipline, namely, the discipline of *Ontology*, in philosophy. *Formal Ontology* has exactly the objective of developing domain-independent theories and systems of categories and their ties that could then be used to articulate conceptualizations in different domains in reality. More recently, the discipline of *Applied Ontology* has developed systematic and repeatable techniques for applying these theories in solving problems in concrete domains[1][6]. Given this essential role played by Ontology in this view of the discipline of Conceptual Modeling, we have termed it elsewhere *Ontology-Driven Conceptual Modeling* [7]. However, exactly due to this dependence, it occurred to us that the term is actually pleonastic. To put bluntly: if conceptual modeling is about representing aspects of the physical and social world and for promoting a shared understanding of this reality among human users [8], then all conceptual modeling should be ontology-driven!

The importance of Ontology as a foundation for Conceptual Modeling is not new in this discipline. There is an established tradition and a growing interest in using ontological theories for analyzing conceptual modeling languages as well as for proposing methodological guidelines for using these languages in the production of ontologically consistent models [3,9]. However, not until much more recently, Ontology has been used not only as an analysis tool but also in the development of engineering tools such as conceptual modeling languages with explicitly defined and properly axiomatized metamodels [10], as well as computational environments supporting automated model verification, validation and transformation [11,12]. These are complexity management tools that are fundamental for addressing the challenge highlighted by Dijkstra's advice. In this paper, I would like to concentrate on a different (albeit complementary and intimately related) set of complexity management tools. The set includes three of these

[1]The relation between Formal and Applied Ontology can be understood in analogy to the relation between Pure and Applied Mathematics.

tools, all related to the notion of patterns, namely: *Ontological Conceptual Patterns*, *Ontological Anti-Patterns*, and *Ontology Pattern Languages*.

The remainder of the paper is organized as follows. In section 2, I briefly discuss the notion of ontological commitment of a language, as well as the notion of foundational ontologies to which general conceptual modeling languages should commit. Section 3 discusses the notion of *Ontological Conceptual Patterns (OCPs)* as methodological mechanisms for encoding basic ontological micro-theories. In that section, I also briefly elaborate on the idea of *Ontology Pattern Languages (OPLs)*, as systems of representation that take OCPs as higher-granularity modeling primitives. In section 4, I elaborate on *Ontological Anti-Patterns (OAP)* as structures that can be used to systematically identify recurrent possible deviations between the set of valid state of affairs admitted by a model and the set of state of affairs actually intended by the stakeholders. In particular, I illustrate here these tools from the point of view of one particular language and ontology. Finally, section 5 presents some final considerations.

2. Ontological Foundations for Conceptual Modeling

Figure 1 below depicts the well-known *Semiotic Triangle*. The dotted line in the base of this triangle between language and reality highlights the fact that the relation between them is always intermediated by a certain *conceptualization*.

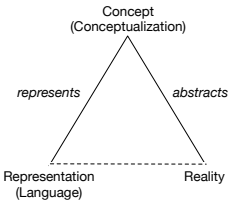

Fig. 1. The Semiotic Triangle

The *represents* relation in Figure 1 stands for the so-called *real-world semantics* of the language, i.e., the function that assigns meaning to the language constructs in terms of elements constituting a conceptualization. This relation also represents the *ontological commitment* of the language [13]. In other words, any representation system that has *real-world semantics* (i.e., which is not limited to purely mathematical formal semantics) has an ontological commitment. As discussed in depth in [13], given this ontological commitment we can systematically evaluate the ontology adequacy of the language, i.e., the adequacy of the language to represent phenomena in reality according to that conceptualization. On one hand, it informs the expected *expressivity* of the language, i.e., that the language should have a maximally economic set of constructs that allows it to represent the distinctions put forth by that conceptualization. On the other hand, it informs the expect *clarity* of the language, i.e., that the language should be such that any valid combination of its constructs should have a univocal interpretation in terms of that conceptualization [3]. However, this ontological commitment does something else of uttermost importance: it informs the set of formal constraints that should be included in the language metamodel to restrict the set of grammatically *valid models of the language* to exact those models that are compatible with that ontological

commitment, i.e., those models that represent state of affairs that are deemed acceptable according to that conceptualization. These are named the *intended models of the language* according to that ontological commitment [13].

As discussed in depth in a number of papers [3, 13], in the case of general conceptual modeling, the ontological commitment of this language should be to a domain independent system of categories and their ties that can be used to articulate conceptualizations of reality in different domains, i.e., a *Foundational Ontology*.

Since our first paper on this topic in this very conference [9], we have engaged in a research program to develop a philosophically sound, formally axiomatized and empirically informed foundational ontology that could serve as a foundation for conceptual modeling. This ontology later termed *UFO (Unified Foundational Ontology)* aggregates results from disciplines such as Analytical Philosophy, Cognitive Science, Philosophical Logics and Linguistics. This ontology is composed of a number of theories addressing the foundation of all classical conceptual modeling constructs including Object Types and Taxonomic Structures, Part-Whole Relations, Intrinsic and Relational Properties, Events, Weak Entities, Attributes and Datatypes, etc. [10, 14-17].

In [10], we have proposed a conceptual modeling language that ontologically commits to this foundational ontology. As we have produced this language through the analysis and redesign of the UML 2.0 metamodel (more specifically, the fragment of UML class diagrams), it later came to be dubbed *OntoUML*. As demonstrated in [10], UML contained many problems of ontological adequacy that needed to be addressed and, in one sense it would have been easier to just define a new conceptual modeling language from scratch. However, UML presented some important features (besides its significant base of users), namely, it had an explicitly defined metamodel coded in a standard metamodeling framework. Building OntoUML by redesigning this metamodel then allowed the language to be used by computational tools that could process implemented metamodels based on MOF (Meta-Object Facility), as well as enable formal verification of OntoUML models with available OCL (Object Constraint Language) tools. We leveraged on this features when building a model-based editor for this language ([11] and, more recently, with continuous updates in http://nemo.inf.ufes.br).

3. OntoUML as an Ontology Pattern Language

Due to the ontological commitment to UFO, the metamodel of OntoUML includes: (i) modeling primitives that reflect ontological distinctions put forth by this ontology; (ii) formal constraints that govern how these constructs can be combined, which are derived from the axiomatization of the ontology. As a result of (ii), we have that the only grammatically correct models that can be produced using OntoUML are those that are consistent with the axiomatization of UFO. However, another consequence of (ii) is that modeling elements of OntoUML never occur freely. In contrast, they only appear in certain modeling configurations and combined with other modeling elements, thus forming certain *modeling patterns*. These patterns are higher-granularity modeling primitives that can be said to represent micro-theories constituting UFO. We term these *Ontology Conceptual Patterns* [18]. In the sequel, I illustrate this idea with some of OntoUML distinctions among different categories of types and relations.

In UFO's theory of types, we have a fundamental distinction between what are named *Sortal* and *Non-Sortal types*. A sortal is a type whose instances obey a uniform principle of identity. A principle of identity, in turn, is a principle with which we can judge if two individuals are the same or, as a special case, what changes an individual can undergo and still be the same. A stereotypical example is the type Person. Contrast it with the type Insurable Item. Whilst in the former case all instance of that type obey the same principle of identity, in the latter case, the type classifies instances of different kinds (e.g., cars, boats, people, houses, body parts, works of art) and that obey different principles of identity. A *Kind* is a sortal which is rigid. Rigidity can be characterized as follows: a type T is rigid iff all instances of that type are necessarily (in the modal sense) instances of that type, i.e., the instances of T cannot cease to be an instance of T without ceasing to exist. In contrast with rigidity, we have the notion of anti-rigidity: a type T' is anti-rigid iff every instance of that type can cease to be an instance of that type (again, in the modal sense), i.e., instances of T' can move in an out of the extension of T' in different possible worlds while maintaining their identity. As formally shown in [10], every object in a conceptual model must obey a unique principle of identity and, hence, must be an instance of a unique kind. As consequence, a sortal T is either a kind or specialize (directly or indirectly) a unique kind.

Among the anti-rigid sortal types, we have again two subcatetories: *Phases* and *Roles*. In both cases, we have that the instances can move in and out of the extension of these types without any effect on their identity. However, while in the case of phases these changes occur due to a change in the intrinsic properties of these instances, in the cases of roles, they occur due to a change in their relational properties. Contrast the types Child, Adolescent, Adult as phases of Person with the roles Student, Husband or Wife. In the former cases, it is a change in intrinsic properties of a person that causes her to move in and out of the extension of these phases. In contrast, a student is a role that a person plays when related to an education institution, and it is the establishment (or termination) of this relation that alters the instantiation relation between an instance of person and the type Student. Analogously, a husband is a role played by a person when married to a (person playing the role of) wife. Thus, besides being anti-rigid, the Role category possesses another meta-property (absent in phases) named *Relational Dependence* [10]. As a consequence, we have that the following constraints must apply to Roles: every Role in an OntoUML conceptual model must be connected to an association representing this relational dependence condition. Moreover, the association end connected to the depended type (e.g., Education Institution for the case of Student, Wife for the case of Husband) in this relation must have a minimum cardinality ≥ 1 [10]. In contrast, phases always occur in the so-called *Phase Partition* of a type T obeying the following constraints: (i) a phase Partition $\langle P_1...P_n \rangle$ defines an actual partition of sortal S , i.e., (i.a) in every situation, every instance of P_i is an instance of S; Moreover, (i.b) in every situation, every instance of S is an instance of exactly one P_i; (ii) for every instance of type S and for every phase P_i in a Phase Partition specializing S, there is a possible world w in which x is not an instance of P_i. This implies that, in w, x is an instance of another Phase P_j in the same partition [10].

The aforementioned ontological constraints defining Roles cause the manifestation of its constructs in OntoUML to obey necessarily the pattern of Figure 2.a: (i) all roles must specialize (directly or indirectly) a unique kind and, hence, must be a directly

specialization of a sortal S; (ii) roles must be connected to a characterizing relation with an opposite association having a minimum cardinality higher or equal to one (symbolizing the relational dependence condition). Likewise, the ontological axioms defining phases cause the manifestation of its construct in OntoUML to obey necessarily the pattern of fig. 2.b.

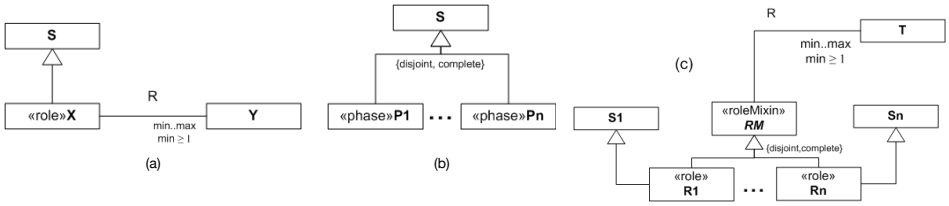

Fig. 2. Role Pattern (a), Phase Partition Pattern (b) and the RoleMixin Pattern (c).

Distinctions generated by the variation of these ontological meta-properties can also be found among non-sortals. One example is the notion of a *RoleMixin*. A RoleMixin is a non-sortal, which is anti-rigid and relationally dependent. In other words, the RoleMixin category is similar to and, hence, is subject to many of the same constraints of the Role category. However, unlike a role, a RoleMixin classify entities that instantiate different kinds (and that obey different principles of identity). Once more, the ontological axioms defining a RoleMixin cause it to manifest in OntoUML necessarily following a particular pattern depicted in Figure 2.c. Like Roles, RoleMixins must be connected to a characterizing relation with an opposite association having a minimum cardinality higher or equal to one (symbolizing the relational dependence condition). However, since RoleMixins classify entities of different kinds, they must be partitioned in a series of specializing sortals (roles), each of which classify entities of a particular kind [10].

Finally, in UFO, we have a fundamental distinction between the so-called *formal* and *material relations*. A formal relation is a relation that holds directly between its relata and that is reducible to intrinsic properties of these relata. Take, for instance, the relation of being-taller-than between people. If John is taller than Paul then this relation is established by the mere existence of John and Paul. Moreover, in this case, there is no real connection between John and Paul, but the relation is reducible to intrinsic properties of these two individuals, namely, John is taller than Paul iff John's height is bigger than Paul's height. Now, take the case of relations such as being-married-to, being-enrolled-at, being-employed-by, being-a-customer-of, etc. These relations are not reducible to intrinsic properties of their relata. In contrast, in order for these relations to hold, something else needs to exist connecting their relata, namely, particular instances of marriages, enrollments, employments and purchases. These mediating entities can be thought as aggregations of relational properties and are termed *relators* [10]. Relations that are founded on these relators are termed *material relations*. As discussed in [10], the explicit representation of relators solves a number of conceptual modeling problems, including the classical problem of the *collapse of cardinality constraints*. Furthermore, as demonstrated in [16], relators also play a decisive role in providing precise methodological guidelines for systematically choosing between the

constructs of *association specialization*, *subsetting* and *redefinition*. Once more, in OntoUML, a material relation appears in a model connected to a relator from which it is derived forming the pattern depicted in Figure 3. In this pattern, the dashed relation is termed derivation and connects a material relation with the relator from which it is derived; the mediation relation is a relation of existential dependence connecting an instance of a relator with multiple entities of which a relator depends (e.g., the marriage between Paul and Mary existentially depends on Paul and Mary; the employment between John and the UN likewise can only exist whilst John and the UN exist). Moreover, the cardinality constraints of the derived material relation and of the derivation relation are constrained by the cardinality constraints of these (otherwise implicit) mediation relations (some of these constraints are illustrated in Figure 3) [10].

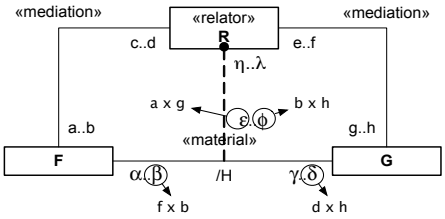

Fig. 3. Relator and Material Relations Pattern.

Since the formal modeling primitives of this language can only appear following these patterns, these patterns end up being the *actual* modeling primitives of the language. As a consequence, modeling in OntoUML is done by the chained application of these ontological patterns [19]. This idea is illustrated in Figure 4. We start by modeling the type Customer. We first identify that a Customer is a RoleMixin: instances of Customer can be different kinds (people and organizations); Customer is an anti-rigid type (no Customer is necessity a Customer); in order for someone to be a Customer, she has to purchase something from a Supplier. In applying the RoleMixin pattern of Figure 2.c, we identify the presence of two phases (Living Person and Active Organization), a role (Supplier, which is assumed to be played by entities of the unique kind Organization) and a relation (purchases from). We then expand this model by applying to phases and roles the patterns of Figure 2.a and 2.b, respectively. Finally, we apply the pattern of Figure 3 to the material relation *purchases from*.

This strategy of building models by the successive instantiation of these patterns has been implemented in the new version of the OntoUML editor. This approach can bring several benefits to conceptual modeling. Firstly, since these patterns are the representation of ontological theories, the construction of models by instantiating these patterns preserves ontological consistency *by construction*. This can also facilitate the process of model building, especially to novice users. The hypothesis is that in each step of the modeling activity, the solution space that characterizes the possible choices of modeling primitives to be adopted is reduced. This strategy, in turn, reduces the cognitive load of the modeler and, consequently, the complexity of model building using this language [19]. Moreover, this strategy also brings more uniformity to the models (which become described in terms of known patterns) and provides for a natu-

ral unit of conceptually breaking down the models in cognitive manageable pieces. However, there is an additional aspect that I would like to highlight here. As previously mentioned, each of these ontological patterns embodies an ontological micro-theory. This means that each application of these patterns implies to the inclusion of a predefined set of formal axioms in the logical rendering of the resulting model (see, for instance, [17]).

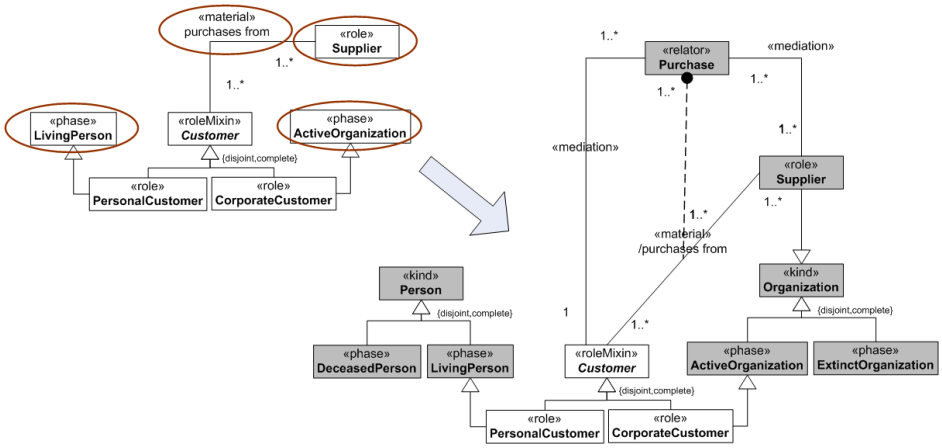

Fig. 4. Model expansion by iterative application of ontological patterns

Up to now, I have focused on patterns that organize the possible manifestations of the modeling primitives of an ontology-based modeling language. Now, I would like to highlight the existence of a second class of conceptual patterns, termed *Analysis Patterns*. These patterns can contribute to conceptual modeling by offering a systematic way of analyzing certain ontological properties of the models. Take for example the much-discussed problem of deciding on the transitivity of part-whole relations. Parthood is non-transitive (i.e., transitive in certain cases and intransitive in others) [10]. This issue is of great importance since transitivity plays a fundamental role both conceptually (e.g., to afford inferences in problem-solving) and computationally (e.g., to afford propagations of properties and events in a transitive chains, as well as automated reasoning with parts). As discussed in [10], precisely identifying the scope of transitivity of part-whole relations requires solving fundamental ontological problems. In [10], using UFO's theory of relations, I have formally proved a number of situations in which part-whole relations should be taken as transitive. Now, the proof presented there demands for its full understanding at least a basic notion of logics and an advanced understanding of formal ontology. Since this obviously compromises the scalability of the proposed solution, [10] also advances a number of *visual patterns* derived from the underlying theory, and that can be directly applied to diagrams to isolate the scope of transitivity of functional part-whole relations (Figure 5). It is important to emphasize that these patterns can be used to isolate the contexts of transitivity in a diagram *regardless of the content of what is being represented there*. As a consequence, fully automated tool support can be built for this task in a relatively simple

way, since the underlying algorithm merely has to check structural (topological) properties of the graph and not the content of the involved nodes. In fact, the automatic identification of these patterns has also been implemented in the OntoUML editor.

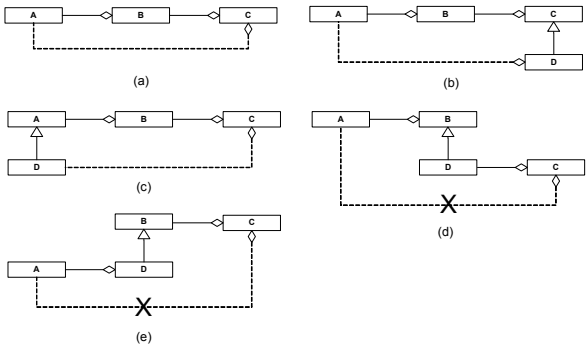

Fig. 5. Patterns for identifying the scope of transitivity of Part-Whole Relations

Figure 6 identifies an instance of the pattern of Figure 5.b. In this model, the relation **A** between Mitral Valve and Musician can be inferred in conformance with this pattern. In contrast, relation **B** between Human Heart and Orchestra cannot be asserted in the model since it actually amounts to a case of the anti-pattern of Figure 5.d.

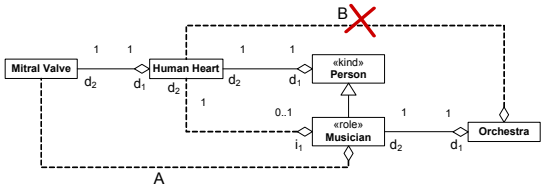

Fig. 6. Example (A) and Counterexample (B) of warranted inference of part-whole relation

4. Ontological Anti-Patterns

By incorporating the ontological constraints of a foundational theory, a modeling language such as the one discussed in the previous section prevents the representation of ontologically non-admissible states of affair in conceptual models represented in that language. However, it cannot guarantee that the produced conceptual models will have as instances only those that represent intended state of affairs. This is because the admissibility of *domain-specific states of affairs* depends on domain-specific rules, not on ontological ones. To illustrate this point, suppose a conceptual model representing a transplant. In this case, we have domain concepts such as Person, Transplant Surgeon, Transplant, Transplanted Organ, Organ Donor, Organ Donee, etc. The model fragment of Figure 7, which models aspects of this domain, does not violate any ontological rule. In fact, this model can be assembled by instantiating instances of the aforementioned *role modeling* and *relator patterns*. However, there are still unintended states of affairs (according to a conceptualization assumed here) that are represented by valid instances of this model. Examples include a state of affairs in which the Donor, the Donee and the Transplant Surgeon are one and the same Person (Figure 8.a), but also

the state of affairs in which the same person plays the roles of Donor and Surgeon (Figure 8.b) or Donor and Donee. Please note that: (a) the model instances of Figures 8a-b are valid instances of the model of Figure 7; (b) these model instances do not represent intended state of affairs according to our assumed conceptualization of the domain of transplants; (c) the state of affairs represented by these model instances are only considered inadmissible (unintended) due to domain-specific knowledge of social and natural laws. Consequently, they cannot be ruled out a priori by a domain independent system of ontological categories.

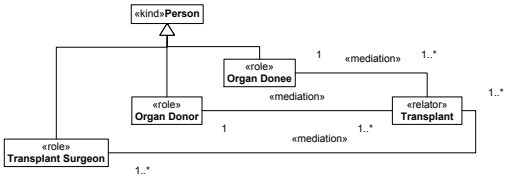

Fig. 7. A fragment of a conceptual model in the domain of organ transplants

Guaranteeing the exclusion of unintended states of affairs without a computational support is a practically impossible task for any relevant domain. In particular, given that many fundamental ontological distinctions are modal in nature, in order to validate a model, one would have to take into consideration the possible valid instances of that model in all possible worlds. In [12], we have proposed an approach for OntoUML that offers a contribution to this problem by supporting conceptual model validation via visual simulation. On the one hand, it aims at proving the finite satisfiability of a given ontology by presenting a valid instance (logical model) of that ontology. On the other hand, it attempts to exhaustively generate instances of the model in a finite scope. The generated model instances confront a modeler with states of affairs that are deemed admissible by the model's current axiomatization. This enables modelers to detect unintended states of affairs and to take the proper measures to rectify the model.

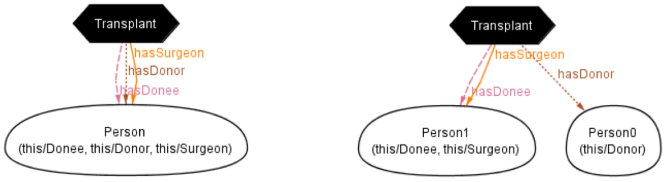

Fig. 8. Examples of valid but unintended instances of the Organ Transplant Model

After running simulations of the model of Figure 7, the conceptual modeler is presented with the consequences of her specification. The set of possible instances of this model, produced automatically by this simulator, includes the two models presented in Figure 8. When faced with a situation in which the Donor, Donee and Surgeon roles are played by the same person, the modeler can realize that the model at hand has been *underconstrained*, and then include a constraint in the model to exclude this unintended situation. Now, suppose the situation in which the modeler tries to rectify this model by declaring the types Transplant Surgeon, Organ Donor and Organ Donee as mutually

disjoint. In a follow up execution of simulating this ontology, she then realizes that it is not possible, for example, for an Organ Donor to receive an organ in a different transplant, and for a Transplant Surgeon to be either an Organ Donor or an Organ Donee in different transplants. When facing this new simulation results, the modeler can realize that the model has been *overconstrained*. After all, there is no problem in having the same person as Organ Donor and Donee, or as Surgeon and Donor (Donee), it is only that the same person cannot play more than one of these roles in the same transplant! (this being the actual formal constraint that should be included in the model). In summary, the idea is that in this multi-step interaction with the model simulator, the modeler can keep refining the domain constraints to increasingly approximate the possible model instances of the model to those that represent admissible states of affairs according to the underlying conceptualization.

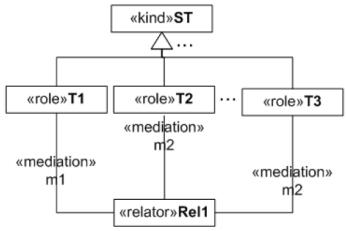

Fig. 9. The RWOR anti-pattern

In [7], we have employed this validation strategy over a benchmark of 52 OntoUML models. In this empirical investigation, we managed to identify model structures that would systematically create deviations between the sets of valid and intended model instances. When these structures appeared in at least roughly 1/3 of the models, we catalogued them as *anti-patterns*. In [7], we have identified 7 of these anti-patterns. In the OntoUML editor, we have implemented a strategy for the automatic detection of these anti-patterns as well as for systematically correcting them via the inclusion of proper formal constraints. For instance, in Figure 7, the problem of model *underconstraining* identified is caused by the manifestation of an anti-pattern termed *RWOR (Relator with Overlaping Roles)*. This anti-pattern (Figure 9) is characterized by a Relator (Rel_1) mediating two or more Roles (T_1, $T_2 \ldots T_n$) whose extensions overlap, i.e., these roles have their identity principle provided by a common Kind as a supertype (ST). In addition, the roles are not explicitly declared disjoint. This modeling structure is prone to be overly permissive, since there are no restrictions for an instance to act as multiples roles for the same relator. The possible commonly identified intended interpretations are that: (i) the roles are actually disjoint (disjoint roles), i.e., no instance of ST may act as more than one role for the same instance of a relator Rel_1 (mutually exclusive roles); (ii) some roles may be played by the same instance of ST, while others may not (partially exclusive roles). An alternative case is: (iii) one in which all or a subset of the roles in question are mutually exclusive but across different relators. An example of this anti-pattern may also be found in the model of Figure 4: a possible instance of that model is one involving more than one supplier, and having the same organization playing both the roles of Customer and Supplier within the scope of the same purchase.

5. Final Considerations

Semantic interoperability will more and more be a pervasive force driving and constraining the development of Information Systems (including Sociotechnical Systems). Information Systems will need to be constructed out of the interconnection of different autonomously developed subsystems and/or will need to be conceived as potential subsystem in multiple yet-to-be conceived larger systems. In this scenario, *conceptual modeling* plays a fundamental role, helping us to understand, elaborate, negotiate and precisely represent subtle distinctions in our multiple conceptualizations of reality. In other words, conceptual modeling should help us to represent proper *"theories of the real-world"* (to use Mealy's expressions) that are both *ontologically consistent* and maximally explicit with respect to their *ontological commitments*. However, in order to successfully play this role, conceptual modeling must rely on sound foundations. Developing these foundations is necessarily an exercise in *Ontology*. Furthermore, since conceptual models are meant to support humans in increasingly complex and interconnected domains, from these foundations, we must develop a number of tools for complexity management. In this paper, I have briefly discussed a particular set of these tools including *Patterns*, *Anti-Patterns* and *Pattern Languages*. There is of course an extensive body of literature on these three topics. However, I focused here on: (i) *Ontological Conceptual Patterns (OCPs)*, i.e., patterns that emerge from the ontological distinctions and axiomatization of foundational ontologies; (ii) *Ontological Pattern Languages (OPLs)*, i.e., systems of representation that take these OCPs as modeling primitives; (iii) *Ontological Anti-Patterns*, i.e., recurrent configurations that potentially make a particular model accept as *valid* some instances that are not *intended* (or, in other words, that are not compatible with its ontological commitment).

I have conducted the discussion here focusing on a particular foundational ontology (UFO) and a particular language based on it (OntoUML). Due to space limitations, I have illustrated my argument using only a very small subset of the patterns and anti-patterns comprising this approach. Additional examples can be found in: [15], in which we present an ontological pattern for decoupling the representation of qualities from the multiple quality spaces on which they can be projected; [17], in which we present a number of patterns derived from a foundational ontology of events. Furthermore, in [14], I formally show how in classical derivation patterns such as *derivation by union* or *derivation by exclusion,* the ontological meta-properties of the derived types can be inferred from the meta-properties of the types participating in the derivation rules.

Finally, there is a very important topic related to ontological patterns and pattern languages that I did not have the chance to discuss here. The modeling patterns discussed in this article are all domain-independent as they are all derived from a domain-independent ontological theory. However, *Domain-Related Ontological Patterns (DROPs)* can also be derived from the so-called *Domain and Core Ontologies*. In particular, as discussed in [20], patterns derived from Core Ontologies can typically be organized in *Domain-Related Ontology Pattern Languages (DROPL)*. In that paper, for instance, we illustrate this approach by developing a DROPL in the domain of Software Processes.

Acknowledgements. I am grateful to G. Wagner, N. Guarino, R. Falbo, R.S.S. Guizzardi, J.P.A. Almeida, J. Mylopoulos and the members of NEMO for many years of fruitful collaboration. This research was partially supported by the Lucretius ERC Advanced Grant # 267856.

References

1. Mealy, G. H., Another Look at Data, AFIPS Conference Proceedings, Volume 31, Washington, DC: Thompson Books, London: Academic Press, 525-534, 1967.
2. Object Management Group, Semantic Information Modeling for Federation (SIMF) Request for Proposals, 2011.
3. Weber, R., Ontological Foundations of Information Systems, Coopers & Lybrand, Melbourne, 1997.
4. Fox, M., A Foundational Ontology for Global City Indicators, Global Cities Institute Working Papers no. 3, 2013.
5. Dijkstra, E.W., The Humble Programmer, Communications of the ACM, 15:10, Oct. 1972.
6. Guarino, N., Musen, M., Applied Ontology: Focusing on Content, Applied Ontology, Vol. 1, pp. 1-5, IOS Press, 2005.
7. Guizzardi, G., Sales, T.P., Detection, Simulation and Elimination of Semantic Anti-Patterns in Ontology-Driven Conceptual Models. Proc. of 33rd International Conf. on Conceptual Modeling (ER 2014), Atlanta.
8. Mylopoulos, J., Conceptual modeling and Telos. In P. Loucopoulos & R. Zicari (eds.), Conceptual Modeling, Databases, and CASE (Chapter 2, pp. 49–68). Wiley, 1992.
9. Guizzardi, G., Herre, H., Wagner G., On the General Ontological Foundations of Conceptual Modeling, 21st International Conf. on Conceptual Modeling (ER 2002), Tampere, 2002.
10. Guizzardi, G.: Ontological foundations for structural conceptual models. Centre for Telematics and Information Technology, University of Twente, The Netherlands, (2005).
11. Benevides, A.B., Guizzardi, G.: A Model-Based Tool for Conceptual Modeling and Domain Ontology Engineering in OntoUML, 11th International Conf. on Enterprise Information Systems (ICEIS), Milan, 2009.
12. Benevides, A.B. et al.: Validating Modal Aspects of OntoUML Conceptual Models Using Automatically Generated Visual World Structures. Journal of Universal Computer Science. 16, 2904–2933, 2010.
13. Guizzardi, G., On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models, Frontiers in Artificial Intelligence and Applications, Databases and Information Systems IV, IOS Press, Amsterdam, 2007.
14. Guizzardi, G., Ontological Meta-Properties of Derived Object Types, 24th International Conference on Advanced Information System Engineering (CAiSE'12), Gdansk, Poland.
15. Guizzardi, G. et al., In the Defense of a Trope-Based Ontology for Conceptual Modeling: An Example with the Foundations of Attributes, Weak Entities and Datatypes, 25th International Conference on Conceptual Modeling (ER'2006), Tucson, 2006.
16. Costal, D., Goméz, C., Guizzardi, G., Formal Semantics and Ontological Analysis for Understanding Subsetting, Specialization and Redefinition of Associations in UML, 30th International Conference on Conceptual Modeling (ER 2011), Brussels, Belgium, 2011.
17. Guizzardi, G. et al., Towards Ontological Foundations for the Conceptual Modeling of Events, 32nd International Conf. on Conceptual Modeling (ER 2013), Hong Kong, 2013.
18. Falbo, R. at al., Ontology Patterns: Clarifying Concepts and Terminology, 4th International Workshop on Ontologies and Semantic Patterns (WOP 2013), Sydney, 2013.
19. Guizzardi, G. et al., Design Patterns and Inductive Modeling Rules to Support the Construction of Ontologically Well-Founded Conceptual Models in OntoUML, 3rd International Workshop on Ontology-Driven Information Systems (ODISE 2011), London, UK.
20. Falbo, F. et al., Organizing Ontology Design Patterns as Ontology Pattern Languages, 10th Extended Semantic Web Conference (ESWC 2013), Monpellier, France.