Detection, Simulation and Elimination of Semantic Anti-patterns in Ontology-Driven Conceptual Models

Giancarlo Guizzardi, Tiago Prince Sales

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

Abstract. The construction of large-scale reference conceptual models is a complex engineering activity. To develop high-quality models, a modeler must have the support of expressive engineering tools such as theoretically well-founded modeling languages and methodologies, patterns and anti-patterns and automated support environments. This paper proposes Semantic Anti-Patterns for ontology-driven conceptual modeling. These anti-patterns capture error prone modeling decisions that can result in the creation of models that allow for unintended model instances (representing undesired state of affairs). The anti-patterns presented here have been empirically elicited through an approach of conceptual models validation via visual simulation. The paper also presents a tool that is able to: automatically identify these anti-patterns in user's models, provide visualization for its consequences, and generate corrections to these models by the automatic inclusion of OCL constraints.

Keywords: Ontology-Driven Conceptual Modeling, Semantic Anti-Patterns

1. Introduction

Conceptual modeling is a complex activity. In [1], an analogy is made between the construction of large reference conceptual models (or reference ontologies) and the programming of large computer systems, referencing the famous E. W. Dijkstra´s ACM Turing lecture entitled "The Humble Programmer". In both cases, we have an acknowledgement of the limitations of the human mind to address the large and fast increasingly intrinsic complexity of these types of activities. For this reason, human conceptual modelers and ontologists should make use of a number of suitable complexity management engineering tools to maximize the chances of a successful outcome in this enterprise. As discussed in [1], among these tools, we have modeling languages and methodologies, patterns and anti-patterns, as well as automated supporting environments for model construction, verification and validation.

In recent years, there has been a growing interest in the use of Ontologically Well-Founded Conceptual Modeling languages to support the construction and management of these complex artifacts. OntoUML is an example of a conceptual modeling 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) [2]. This language has been successfully employed in a number of industrial projects in several different domains, such as Petroleum and Gas, Digital Journalism, Complex Digital Media Management, Off-Shore Software Engineering, Telecommunications, Retail Product Recommendation, and Government. Besides the modeling language itself, the OntoUML approach also offers a model-based environment for model construction, verbalization, code generation, formal verification and validation [3]. In particular, the validation strategy employed there makes use of an approach based on visual model simulation [4]. In this paper, we make use of this approach for eliciting *anti-patterns*.

An anti-pattern is a recurrent error-prone modeling decision [5]. In this paper, we are interested in one specific sort of anti-patterns, namely, model structures that, albeit producing syntactically valid conceptual models, are prone to result in unintended domain representations. In other words, we are interested in configurations that when used in a model will typically cause the set of valid (possible) instances of that model to differ from the set of instances representing *intended state of affairs* in that domain [2]. We name here these configurations *Semantic Anti-Patterns*.

The contributions of this paper are two-fold. Firstly, we contribute to the identification of Semantic Anti-Patterns in Ontology-Driven Conceptual Modeling. We do that by carrying out an empirical qualitative approach over a model benchmark of 52 OntoUML models. In particular, we employ the visual simulation capabilities embedded in OntoUML editor [3]. Secondly, once these anti-patterns have been elicited, we extend the OntoUML editor with a number of features for: (a) automatically and proactively detecting anti-patterns in user models; (b) supporting the user in exploring the consequences of the presence of an anti-pattern in the model and, hence, deciding whether that anti-pattern indeed allows for unintended model instances; (c) automatically generating OCL constraints that excluded these unintended model instances.

The remainder of this paper is organized as follows: in Section 2, we briefly elaborate on the modeling language OntoUML and some of its ontological categories, as well on the approach for model validation via visual simulation embedded in the OntoUML editor; Section 3 characterizes the model benchmark used in this research; Section 4 presents the elicited Semantic Anti-Patterns with their undesired consequences and possible solutions; section 5 elaborates on the extensions implemented in the OntoUML editor taking into account these anti-patterns. Finally, Section 6 presents some final considerations of this work.

2. Model Validation via Visual Simulation in OntoUML

The OntoUML language meta-model contains: (i) elements that represent ontological distinctions prescribed by the underlying foundational ontology UFO; (ii) constraints that govern the possible relations that can be established between these elements reflecting the axiomatization of this underlying ontology. These two points are illustrated below using some ontological distinctions among the categories of object types (**Kind**, **Subkind** and **Roles**), trope types (**Relator**) and relations (**formal relations** and **material relations**). For an in depth presentation, formal characterization and empirical

evidence for a number of the ontological categories underlying OntoUML, the reader is referred to [2].

In a simplified view we can state that: Kinds and Subkinds are types that aggregate all the essential properties of their instances and, for that reason, all instances of a given Kind/Subkind cannot cease to instantiate it without ceasing to exist (a meta-property known as *rigidity*). A Kind defines a uniform principle of identity that is obeyed by all its instances; Subkinds are rigid specializations of a Kind and inherit that principle of identity supplied by that unique subsuming Kind. A Role, in contrast, represents a number of properties that instances of a Kind have contingently and in a relational context. A stereotypical example can be appreciated when contrasting the Kind *Person*, the Subkinds *Man* and *Woman* (specializing Person) and the Role *Student* (also specializing Person).

A Relator is the objectification of a relational property (i.e., a complex relational trope) and is intimately connected to an event in which roles are played. Relators are existentially dependent on a multitude of individuals, thus, mediating them [2]. In other words, a relation of **mediation** is a particular type of existential dependence relation connecting a relator to a number of relata. Examples of relators are *Enrollments, Employments, Covalent Bonds* and *Marriages*. Relators are the foundation and truthmakers of the so-called material relations in the way, for instance, that the marriage between John and Mary founds (is the truthmaker of) the relation *is-married-to* between John and Mary (but also the relations *being-the-husband-of*, *being-the-wife-of*), or in the way that the *Enrollment* between Mick and the London School of Economics founds the relation *studies-at* between these two individuals. Contrary to material relations, formal relations hold directly between entities without requiring any intervening (connecting) individual. Examples include the relations of existential dependence and parthood but also *being-taller-than* between individuals.

Regarding characteristic (i) above, OntoUML incorporates modeling constructs that represent all the aforementioned ontological categories (among many others) as modeling primitives of the language. Regarding (ii), the meta-model embeds constraints that govern the possible relations to be established between these categories. These constraints are derived from the very axiomatization of these categories in the underlying foundational ontology. Examples include (among many others): a Role (as well as a Subkind) must be a subtype of exactly one ultimate Kind; a role cannot be a super-type of a Kind or a Subkind; a relator must bear mediation relations to at least two distinct individuals.

As a result of these constraints, as discussed in [2], the only grammatically correct models of OntoUML are ontologically consistent models. In other words, by incorporating ontological constraints in its meta-model, OntoUML proscribes the representation of ontologically non-admissible states of affairs in conceptual models represented in that language. However, as discussed in [5], the language cannot guarantee that, in a particular model, only model instances representing *intended state of affairs* are admitted. This is because the admissibility of domain-specific states of affairs is a matter of factual knowledge, not a matter of consistent possibility [1].

To illustrate this point, we will use for the remainder of the paper the running example presented in Fig.1. This model describes people's *roles* and relevant properties

in the context of a criminal investigation. Some of roles may be the detectives that investigate the crime, other the suspects of committing the crime, but also witnesses that are interrogated by the detectives about the crime. Each investigation has a detective who is responsible for it. Detectives are ranked as officers and captains. Finally, since other relational properties are relevant in investigations, the model also represents parenthood and acquaintance ("person knows person") relations among people.

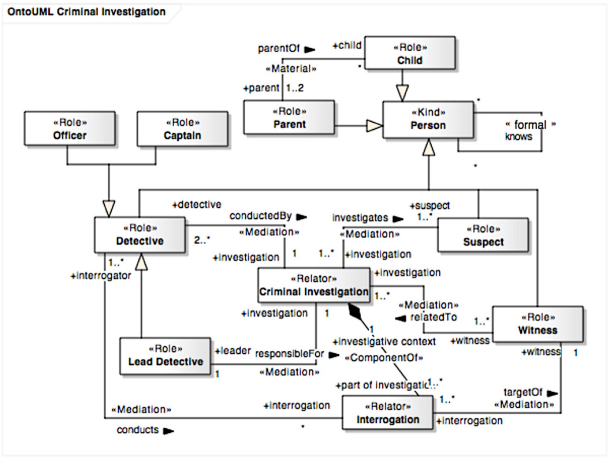

Fig. 1. Partial OntoUML model of the domain of criminal investigation.

Capturing aspects of this domain, the model of Fig.1 does not violate ontological rules; it would have done so, for example, had we placed Suspect as a super-type of Person, or had we represented the possibility of a Suspect or Witness without being related to Criminal Investigation (we assume here a suspect is a suspect in the context of an investigation and so is a witness) [2]. These cases can be easily detected and proscribed by an editor such as the one proposed in [3]. One example is one in which the Lead Detective of an investigation is also a Suspect on that investigation. Another example is one in which a Detective interrogates himself. A third one is one in which someone is his own parent (or a parent of one of her parents). This simple and relatively small model fragment actually contains 13 cases of what we term Semantic Anti-Patterns, i.e., model fragments that when used, typically create a deviation between the set of possible and the set of intended state of affairs [2]. We will return to this point in sections 4 and 5.

Guaranteeing the exclusion of unintended states of affairs without a computational support is a practically impossible task for any relevant domain [1]. 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 [4], the authors propose an automated approach for OntoUML that offers a contribution to this problem by supporting conceptual model validation via visual simulation. In the proposed tool, the models are translated into Alloy [6], a logic language based on set theory, which is supported by an analyzer that, given a context, exhaust-

ively generates possible instances for a given specification and also allows automatic checking of assertions' consistency. The generated instances of a given conceptual model are organized in a branching-time temporal structure, thus, serving as a visual simulator for the possible dynamics of entity creation, classification, association and destruction. In [4], the modeler is then confronted with a visual representation of the snapshots in this world structure. These snapshots represent model instances that are deemed admissible by the ontology's current axiomatization. This enables modelers to detect unintended model instances (i.e., model instances that do not represent intended state of affairs) so that they can take the proper measures to rectify the model.

The comparison between admissible model instances, generated by the Alloy Analyzer, and the intended ones, obtained from domain experts or the conceptual model documentation, highlights possibly erroneous modeling decisions. The recording and categorization of these decisions for a set of OntoUML conceptual models served as a basis for identifying the semantic anti-patterns proposed in this paper. The process for empirically uncovering these anti-patterns is explained in section 3 below.

3. Empirically Uncovering Semantic Anti-Patterns

The approach used in this work for the identification of the proposed set of anti-patterns was an empirical qualitative analysis. The idea was to simulate existing OntoUML conceptual models by employing the approach described in section 2. In a preliminary analysis reported in [7], we studied the recurrence of these anti-patterns across: (i) different domains; (ii) different levels of modeling expertise in Ontology-Driven Conceptual Modeling; (iii) models of different sizes, maturity and complexity.

In that study, we have first started with 9 models selected across the following areas: (1) a Conceptual Model that describes a Brazilian Health Organization; (2) a Conceptual Model that describes the Organizational Structure of Brazilian Federal Universities; (3) a Conceptual Model that describes a Domain of Online Mentoring Activities; (4) an Ontology representing the domain of Transport Optical Network Architectures; (5) an Ontology in the Biodiversity Domain; (6) a Heart Electrophysiology Reference Ontology; (7) an Ontology in the Domain of Normative Acts; (8) an Ontology of Public Tenders; (8) an Ontology in the Domain of Brazilian Federal Organizational Structures.

Regarding levels of expertise, we have classified as "beginners", those modelers with less than one year of experience with OntoUML and its foundations. In contrast, we classified as "experienced", those modelers that had worked with the language for two or more years and had applied the language in large-scale complex domains. In all the analyzed cases, the modelers involved in the creation of the models had a significant experience in traditional conceptual modeling approaches. In our first sampling of models, we had 4 models created by beginners (models 1-3, 9) and 5 models created by experienced modelers (4-8).

Finally, regarding scale and complexity, three of the investigated models were graduate final assignments (models 1-3), two of which were produced by modelers with vast experience in the respective domains (1-2); model (4) was produced by experienced modelers in an industrial project. Moreover, the modelers had access to domain experts as well as a supporting international standard of the domain (ITU-T G.805).

Finally, the resulting ontology was published in a relevant scientific forum in the area of Telecommunications; Model (5) was developed in the Brazilian National Center for Amazon Research in collaboration with domain experts; Model (6) was published in a renowned international journal in the area of Bioinformatics in a special issue of Biomedical ontologies; Models (7-8) were produced in a large-scale industrial project for the Brazilian Regulatory Agency for Land Transportation (ANTT). The modelers had constant access to normative documentation and to domain experts; finally, model (9) was produced by a group of modelers in the Brazilian Ministry of Planning. The group was formed by experts in the domain who had a professional-level experience in traditional conceptual modeling. The size of these models varied from 15-31 classes (between 7-30 associations) for the models produced by beginners (models 1-3, 9) to 46-194 classes (between 29 to 122 associations) for those models produced by experienced researchers.

In what follows, we describe our strategy for identifying anti-patterns across this sample of models. For each of these cases, we started by simulating the model at hand using the approach described in the previous section. This process resulted in a number of *possible model instances* for that model (automatically generated by the Alloy Analyzer). We then contrasted the set of possible instances with the set of *intended instances* of the model, i.e., the set of model instances that represented intended state of affairs according the creators of the models. When a mismatch between these two sets was detected, we analyzed the model in order to identify which structures in the model were the causes of such a mismatch. Finally, we catalogued as anti-patterns those model structures that recurrently produced such mismatches, i.e., modeling patterns that would repeatedly produce model instances that were not intended ones. To be more precise, we considered as anti-patterns the error prone modeling decisions, which occurred in at least one third of the validated models. We carried out this simulation-based validation process with a constant interaction with the model creators (when available), or by inspecting the textual documentation accompanying the models otherwise.

In this first empirical study, we manage to identify 6 initial semantic anti-patterns. The occurrence of these anti-patterns in the studied models was 33.33% for two of the patterns (i.e., the anti-patterns appeared in 1/3 of the models), 66.67%, 77.78%, 88.89% and 100% (i.e., one of the anti-patterns appeared in all the analyzed models). The details of this studied are found in the following preliminary report [7].

This initial study gave us confidence that the adopted method could be used as a means for detecting these semantic anti-patterns. In the follow up study reported here, we manage to assemble a much larger benchmark of 52 OntoUML models. These models can be characterized as follows: (a) 61,53% of the models were produced by experienced modelers while 38,46% of them were produced by beginners; (b) the majority of these models (69,23 %) were graduated assignments at the master and PhD level produced as a result of a 60-hours OntoUML course. We have also that 21,15% of these models were results of graduate dissertations (MSc and PhD thesis) in areas such as Provenance in Scientific Workflow, Public Cloud Vulnerability, Software Configuration Management, Emergency Management, Services, IT Governance, Organizational Structures, Software Requirements, Heart Electrophisiology, Amazonian

Biodiversity Management, Human Genome. Finally, 7,69% of these models were produced in industrial projects in areas such as Optical Transport Networks, Federal Government Organizational Structures, Normative Acts, and Ground Transportation Regulation; (c) in terms of size and complexity, these models varied from a simple model in a graduation assignment containing 11 classes and 14 associations to an industrial model in the domain of Ground Transportation Regulation containing 3775 classes, 3566 generalization relations, 564 generalization sets and 1972 associations. The average number of classes and relations (generalization relations plus associations) when considering all models is 114.92, 169.26, respectively. If only industrial projects are considered these averages go up to 1.027 for classes and 388 for relations.

In order to analyze this new benchmark, we have implemented a set of computational strategies to automatically detect occurrences of these anti-patterns in OntoUML models (see discussion in section 5). By running these algorithms for our initial set of anti-patterns under this benchmark, we managed to refine and extend the initial set elicited in [7] to a refined set of anti-patterns. Table 1 below reports on these new anti-patterns whose automatic strategy of detection and correction have been incorporated in the OntoUML computational editor (see section 5). Among the anti-patterns presented in this new set, one of them is a refinement and extension of the existing STR anti-pattern, two of them are newly discovered anti-patterns, namely, TRI and RWOR. Moreover, RWOR have been generalized under the category RelOver together with the RBOS anti-pattern. Finally, the so-called PA (Pseudo-AntiRigid) anti-pattern from our original catalog was excluded from the analysis conducted here due to the fact that the detection of its occurrences cannot be performed algorithmically.

It is important to highlight that given the size of this new set of models, unlike in our previous study, we were not able to check for each occurrence of these anti-patterns (3612 occurrences!) whether they were always cases of model fragments that entailed unintended consequences. For this reason, in the analysis reported in table 1, each occurrence of an anti-pattern does not necessarily mean an unintended occurrence of the corresponding model fragment. However, in our previous empirical study, we could observe a very strong correlation between the high occurrence of these anti-patterns as model fragments and cases in which they were identified as unintended. In fact, that is exactly why they were identified as anti-patterns (as opposed to purely syntactic constraints) in the first place.

The anti-patterns represented in table 1 are discussed in section 5.

Semantic Anti-Patterns (SAP)	% of occurrences across models	Total # of occurrences
RS	46,15%	1435
IA	71,15%	725
AC	51,92%	155
RelOver (RWOR + RBOS)	30,7%	437
TRI	55,77%	685
BinOver (incl. STR)	48,07%	175

Table. 1. Occurrences of Semantic Anti-patterns in the model Benchmark used.

4. A Catalogue of Semantic Anti patterns

4.1 Relation Specialization (RS)

As depicted in **Fig. 2(a)**, the RS anti-pattern is characterized by the representation of a relation R between two types T1 and T2, such that their respective super-types ST1 and ST2 are also associated by a relation SR. It is important to highlight that ST1 and ST2 are not necessarily direct super-types (depicted in the figure by the sign "…") but also that they are not necessarily strict (proper) super-types. In fact, we can have cases in which T1 = ST1, T2 = ST2, and even a case in which T1 = T2 = ST1 = ST2, i.e., a case in which the model of fig 2(a) degenerates into a model with one type and two type-reflexive relationships R and SR between instances of this unique type. What we have found in our analysis is that there is usually some sort of constraint between R and SR overlooked by the modeler. The solution for eliminating this potential source of problem (in the case the modeler in fact judges this to be one) is to include constraints on the relation between R and SR, thus, declaring R to be either a specialization, a subset, a redefinition or disjoint with relation SR. OntoUML has an in depth treatment of these relations as well as precise ontological guidelines for differentiating when each of these modeling alternatives should be used. An example of an occurrence of this pattern in Fig.1 is the following: we should guarantee that the Lead Investigator responsible for a Criminal Investigation is one of the Detectives conducting that Investigation.

4.2 Relation Between Overlapping Subtypes (RBOS)

The RBOS anti-pattern occurs in a model having two potentially overlapping (i.e., non-disjoint) types T1 and T2 whose principle of identity is provided by a common Kind ST, and such that T1 and T2 are related through a formal relation R as depicted in **Fig. 2(b)**. This problem frequently appears when T1 and T2 are roles, although it can also occur having T1 and T2 as subkinds. The problem here comes from the fact that an object may instantiate both T1 and T2 simultaneously. Occasionally, roles in relation R are played by entities of the same kind ST. However, it is frequently undesired that these roles are played by the same instance of ST. In case T1 = T2 and R is a binary relation this anti-pattern degenerates to a case of the *BinOver* pattern and, in the limit case in which T1, T2 and ST are identical, it degenerates to a particular case of BinOver termed *STR*. BinOver and STR are discussed in the sequel. Moreover, when the relation R in Fig. 2(b) is a material relation (not-necessarily a binary one), this anti-pattern configures a case of the RWOR anti-pattern explained in section 4.6. Possible rectifications of this anti-pattern include characterizing R as (non/anti)reflexive, (non/a)symmetric or (in/anti)transitive, or defining T1 and T2 as disjoint. For example, in Fig.1, we have the formal relation *parentOf* between the roles Child and Parent. Although the two roles in this relation must be played by instances of the same Kind (Person), they cannot be played by the same instance of Person for the same instance of the relation. In this case, the types T1 and T2 are not disjoint, since the same individual can play both these roles (i.e., someone can be a father of person x and son of person y) but not in the same relation instance. In this case, the solution is to declare relation *parentOf* as anti-reflexive, asymmetric and anti-transitive.

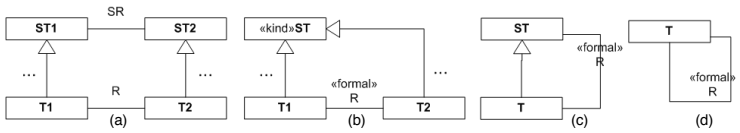

Fig. 2. Structural configuration illustrating **the (a)** RS, **(b)** RBOS, **(c)** BinOver, and **(d)** STR.

4.3. BinOver and Self-Type Relationship (STR)

The BinOver pattern occurs when the two association ends of a binary relations R can be bound to the same instance. A particular case of BinOver is the so-called Self-Reflexive *STR* anti-pattern. BinOver is configured by a relation R having one of its associations ends connected a type T and another of its association ends connected to a super-type ST of T as depicted in **Fig. 2(c)**. In case that ST=T, we have a particular case of BinOver termed Self-Type Relationship (STR) (see **Fig. 2(d)**). Type-Reflexive relations as they appear in these two configurations are usually overly permissive and typically should be constrained using the meta-properties that precisely characterize a formal binary relation such as (in/anti)transitive, (a/non)symmetric, (non/co)reflexive, total, trichotomous or euclidean. This anti-pattern occurs when R are formal relations. In case R is a material relation, this anti-pattern configures a case of the RWOR anti-pattern explained in section 4.6. In Fig. 1, this configuration appears in the relation "knows" between People. Notice that this relation is indeed a formal relation, since it can be reduced to an intrinsic property of the relata (in this case the knowledge of the knowers). In this domain, this relationship is reflexive, asymmetric (but not anti-symmetric) and intransitive (but not anti-transitive). In other domains, e.g., some social networks, this relation can in contrast be considered to be symmetric and transitive.

4.4 Association Cycle (AC)

This anti-pattern consists of three or more types *T1…Tn* connected through an association chain *R1,2…Rn-1,n* (where Rij connects type Ti with type Tj) in a way to form a cycle. In **Fig. 3.(a)**, T1, T2 and T3 form a cycle through the associations R1, R2 and R3. The possible constraints to be applied over this configuration are that these cycles should be reinforced to be either closed or open cycles. A OCL-like constraint having T1 as a reference (i.e., as an OCL context) for the case of closed cycles has the form (*self.T2.T3…Tn.T1.asSet()=self.asSet()*) and for the open cycle the form (*self.T2.T3…Tn.T1->excludes(self)*). In section 5, we discuss in detail an example from Fig.1 in which a close cycle must be guaranteed, namely, that a detective who conducts an interrogation that is part of an investigation must be one of the detectives of that investigation.

4.5. Imprecise Abstraction (IA)

As depicted in **Fig. 3(b)**, this anti-pattern is characterized when two types T1 and T2 are related through an association R with an upper cardinality in both ends greater than one, and at least one of the related types containing its own subtypes. The source of the inconsistency comes from the representation of a single, more abstract association

between T1 and T2, instead of more concrete ones between T1 and T2's subtypes. In this case, there might be domain-specific constraints missing in this model referring to which subtypes of T2 an instance of T1 may be related. As an example, suppose that in **Fig.3(b)** an instance of T1 can only be related through relation R to instances of a particular STi, or that instances of T1 are subject to different cardinality constraints on R for each of the different subtypes STj. An example in the model of Fig.1 is the following: although a Criminal Investigation can have at least two Detectives, exactly one of them must be a Captain.

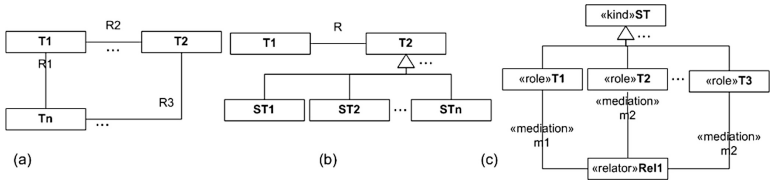

Fig. 3. Structural configuration illustrating **the (a)** AC, **(b)** IA and **(c)** RWOR.

4.6 Relator With Overlapping Roles (RWOR)

The generic structure of the Relator With Overlapping Roles (RWOR) anti-pattern is depicted in **Fig. 3(c)**. It is characterized by a *Relator* (R1) mediating two or more *Roles* (T1, T2… Tn) whose extensions overlap, i.e. have their identity principle provided by a common *Kind* as a super-type (ST). In addition, the roles are not explicitly declared disjoint. This modeling structure is prone to be overly permissive, since there are no restriction for an instance to act as multiples roles for the same relator. The possible commonly identified intended interpretations are that: 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 Rel1 (mutually exclusive roles); some roles may be played by the same instance of ST, while others may not (partially exclusive roles). An alternative case is one in which all or a subset of the roles in question are mutually exclusive but across different relators. An instance of RWOR in our running example is discussed in section 5.

4.7 Twin Relator Instances (TRI)

This anti-pattern occurs when a relator is connected to two or more «mediation» associations, such that the upper bound cardinalities at the relator end are greater than one. The problem associated with this anti-pattern is that it opens the possibility for two distinct instances of the same relator type to co-exist connecting the very same relata instances. We empirically found that the existence of these relator instances in this situation should frequently be subject to several different types of constraints. For instance, it can the case that there cannot be two different relator instances of the same type connecting the very same relata. An example in the domain depicted in Fig.1 could be: one cannot be the subject of a second criminal investigation as a suspect and be investigated by the same detectives that interrogate the same witnesses. There can be cases that the existence of these multiple relators instances are allowed but not sim-

ultaneously (e.g., a passenger can have more than one reservation for the same hotel but not for the same time period). In fact, there can be a number of variations of the cases above due to domain-specificity: (a) two or more relators of the same type can bind the same relata but these relators have to exist separated by a specific time interval from each other (e.g., contracts between the same employee and the same public institution can exist but only if separated by at least two-years from each other), or they can partially overlap but cannot be totally synchronized, etc.; (b) two or more relators of the same type have to vary in at least a specific subset of its roles (e.g., an employee can have more than one valid contract with the same employer at intersecting time intervals, however, not for the same position).

5. Anti-Pattern Detection, Analysis and Elimination

In order to support the approach presented in Section 4, we developed a suite of plugins for validating OntoUML models that have been incorporated in the OntoUML editor (**Fig.4**). With the goal of supporting the entire process described in section 2, this tool supports a set of tasks. First, it allows for the automatic detection of anti-patterns in the model. Since we cannot know a priori which are the unintended situations (if any) that should be excluded from the model, the tool offers a visual simulation environment implementing the approach previously discussed. Finally, when the expert identifies the unintended instances to be excluded, the tool offers semi-automatic correction via the automatic generation of OCL constraints.

In what follows, in order to illustrate this process we use the domain model of Criminal Investigation depicted in Fig.1. Once a model is constructed or loaded into the OntoUML editor (**Fig.4.1**), the anti-pattern detection algorithms embedded in this tool can be activated (**Fig.4.2**). When analyzing the criminal investigation model of Fig.1, the detection algorithms identified 13 candidate occurrences of semantic anti-patterns (**Fig.4.3**): 1 occurrence each of RBOS and STR; 2 occurrences each of *AC*, *RS*, *RWOR* and *TRI*, and 3 occurrences of *IA*. In the following, due to lack of space, we elaborate an example of *AC*, and an example of RWOR in this model.

One identified *AC* is a cycle composed by Criminal Investigation, Detective, Interrogation and, again, Detective (with the respective associations). This possible occurrence of an anti-pattern is shown in the window depicted in **Fig.4.4**. In that window, the modeler can select the option of visualizing possible instances of the model in which the identified anti-pattern is manifested (button "execute with Analyzer"). One visual representation of an instance of this model produced with this functionality is depicted in **Fig.5**. In this instance, detective *Object9* conducts interrogation *Property7*, which is part of the Criminal Investigation *Property2*. However, *Object9* is not one of the detectives conducting Criminal Investigation *Property2*. In other words, the model allows for a representation of a state of affairs in which an interrogation that is part of a criminal investigation is conducted by a detective that is not part of that investigation. Let us suppose that the creators of that model do not intend such a state of affairs. The modelers can then request the editor for an OCL solution that would proscribe instances with this detected unintended characteristic (button "OCL solution" in Fig.4.4). In this case, the OCL constraint to be incorporated in the model (Fig.4.5) is the following:

```
context CriminalInvestigation
inv closedCycle:
self.interrogation.interrogator.investigation->asSet()= self->asSet()
```

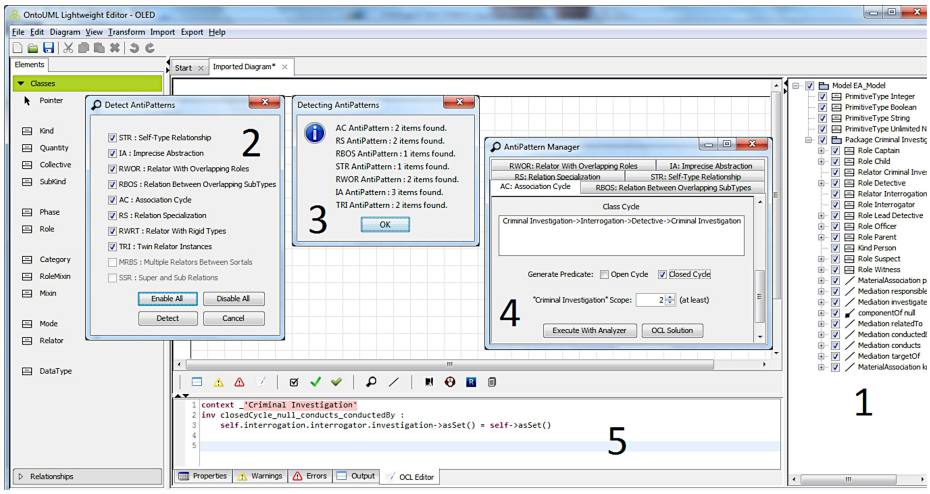

Fig. 4. Anti-Pattern detection and analysis capabilities incorporated in the OntoUML editor

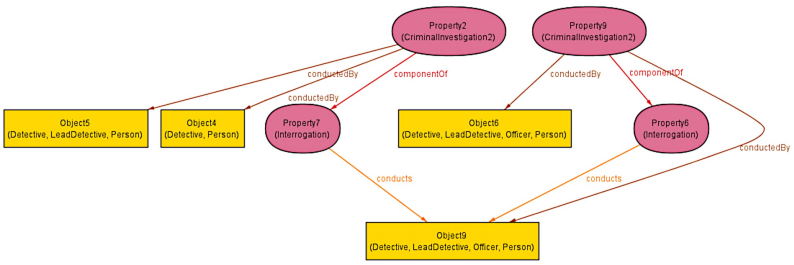

Fig.5. Possible interpretation of the *AC* identified in the Criminal Investigation model.

An example of an identified RWOR anti-pattern involves criminal investigation as a relator that mediates the Roles Detective, Lead Detective, Suspect and Witness. As explained in Section 4, there are three types of possibly unintended instances that can be allowed by an occurrence of this anti-pattern. First, all roles are exclusive in the scope of a particular relator, which means that in each particular investigation the roles of suspect, witness, detective and lead detective are necessarily all instantiated by different people. Second, it may be the case that only some of these roles are exclusive in the scope of a particular relator, for example, the detective and the suspect are exclusive, but not detective and witness, or suspect and witness. Finally, it may also be the case that some of the roles are disjoint (across different relators). For example, suppose the constraint that detectives who participate in an ongoing investigation cannot be considered suspects in another investigation. Let us suppose that as a first action to rectify the model the modeler chooses to declare all roles as exclusive w.r.t. a given investigation. The set of instances of the resulting model, hence, includes the one de-

picted in **Fig.6**. By inspecting the model of Fig.6, the modeler can then realize that she perhaps over-constrained the model since, as a result of declaring all roles as exclusive, we have that the responsible for a given investigation (i.e., the lead detectives) is not considered as a participant of that investigation (i.e., one of its detectives). The modeler can then rectify the model again by choosing among a set of other alternative OCL solutions offered by the OntoUML editor. In **Fig.7**, we show the case in which the modeler chooses both to declare the roles of witness and suspect disjoint w.r.t. a given investigation (constraint on line 1 of the OCL code), as well as the roles witness and detective (line 4), but also to declare that the roles of detective and suspect should be disjoint across different investigations (line 7).

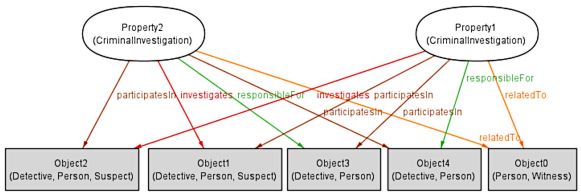

Fig. 6. Exclusive view of the roles in a criminal investigation.

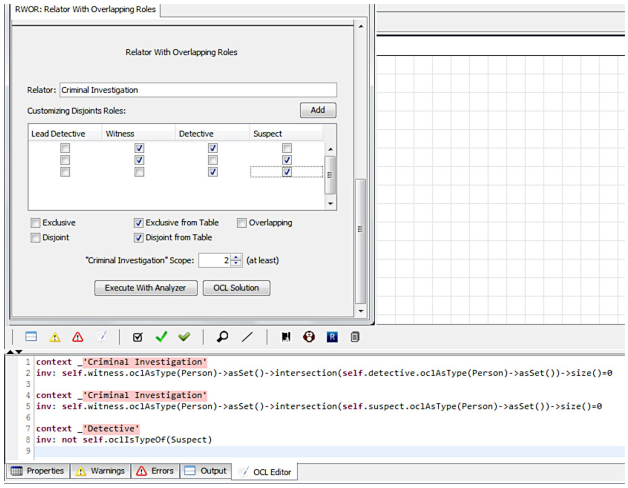

Fig.7. Automatically generated OCL solutions to excluded unintended instances of RWOR

6. Final Considerations

This paper makes a contribution to the theory and practice of ontology-driven conceptual modeling by: (i) presenting a number of empirically elicited Semantic Anti-patterns that were identified as recurrent in a benchmark of conceptual models; (ii) presenting a computational environment that automates the process of supporting detection of anti-patterns, exploration of their consequence in individual models, formal rectification via the inclusion of pre-defined formal constraints. This computational environment is available in https://code.google.com/p/ontouml-lightweight-editor/.

Our approach is in line with authors both in the conceptual modeling and ontology engineering literature. Two representative examples of works in this area are [8] and [9], which discuss methods of detecting anti-patterns in OWL specifications via SPARQL queries. Although sharing the same general objective, our approach differs from these works in a number of important ways. Firstly, our approach is based on a much richer modeling language from the ontological point of view. As a consequence, the anti-patterns addressed by our approach are able to address more subtle ontological conditions such as, for example, the ones involving modality, identity principles as well as a richer ontology of material relations. Secondly, different from these approaches, our method does not aim at detecting general cases involving typical logical misunderstandings. In contrast, it focuses exactly on those cases that cannot be casted as modeling (grammatical) errors by the process of formal verification, and aims at identifying recurrent potential deviations between the sets of valid and intended model instances. Thirdly, for instance in [9], the identified anti-patterns are cases believed to be caused by the lack of modeling experience [9]. Here, as shown by our empirical study, these anti-patterns are recurrent even in models produced by experience researchers. In fact, in pace with [1], we believe that the repeated occurrence of these anti-patterns is an intrinsic feature of the disparity between the increasing complexity of our reference conceptual models and our limited cognitive capacities for dealing with that. Finally, in contrast with these approaches, besides automatic anti-pattern detection, our approach presents a computational environment for model analysis (via visual simulation) and systematic conceptual model rectification.

Acknowledgements. The authors are grateful to João Paulo Almeida, John Guerson and Pedro Paulo Barcelos for fruitful discussions in the topics of this article. This research was partially supported by the Lucretius ERC Advanced Grant # 267856.

References

1. Guizzardi, G.: Theoretical foundations and engineering tools for building ontologies as reference conceptual models. Semantic Web Journal. 1, 3–10 (2010).
2. Guizzardi, G.: Ontological foundations for structural conceptual models. Centre for Telematics and Information Technology, University of Twente, The Netherlands, (2005).
3. Benevides, A.B., Guizzardi, G.: A Model-Based Tool for Conceptual Modeling and Domain Ontology Engineering in OntoUML, 11th ICEIS, Milan (2009).
4. 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).
5. Koenig, A.: Patterns and Anti-Patterns. J. of Object-Oriented Programming. 8 (1995).
6. Jackson, D.: Software Abstractions: Logic, Language, and Analysis. The MIT Press, Cambridge, Massachusetts (2012).
7. Sales, T.P., Barcelos, P.P.F., Guizzardi, G.: Identification of Semantic Anti-Patterns in Ontology-Driven Conceptual Modeling via Visual Simulation. 4th International Workshop on Ontology-Driven Information Systems (ODISE), Graz, Austria (2012).
8. Vrandečić, D.: Ontology Validation, PhD Thesis, University of Karlsruhe (2010).
9. Roussey, C. et al.: SPARQL-DL queries for Antipattern Detection. Workshop on Ontology Patterns. CEUR-WS.org, Boston, USA (2012).