

Chapter 1

Anti-patterns in Ontology-driven Conceptual Modeling: The Case of Role Modeling in OntoUML

Tiago Prince Sales

Department of Information Engineering and Computer Science,
University of Trento, Italy

Giancarlo Guizzardi

Ontology and Conceptual Modeling Research Group (NEMO),
Federal University of Esp rito Santo, Brazil

1.1. Introduction

Given the increasing complexity of ontology-driven conceptual modeling and ontology engineering, there is an urging need for developing a new generation of complexity management tools for these disciplines [12]. These include a number of methodological and computational tools that are grounded on sound ontological foundations. In particular, we should advance in these disciplines a well-tested body of knowledge in terms of Ontology Patterns, Ontology Pattern Languages and Ontological Anti-Patterns . This chapter focuses on the latter.

An anti-pattern is a recurrent error-prone modeling decision [15]. 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 [11]. We name these configurations *Anti-Patterns for Ontology-Driven Conceptual Modeling*, or simply, *Ontological Anti-Patterns*.

In this chapter, we focus on the study of Ontological Anti-Patterns in a particular conceptual modeling language named OntoUML [11]. 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) [11]. UFO is an axiomatic formal theory based on theories from Formal Ontology in Philosophy, Philosophical Logics, Cognitive Psychology and Linguistics. OntoUML 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 [12].

This chapter can be seen as complementary to our earlier work [23]. However, while previously we have focused on anti-patterns that are connected to the modeling of relations, here we focus on anti-patterns that emerge in connection to the modeling of *role types*, i.e., anti-rigid and relationally dependent types [11]. Stereotypical examples of role types include husband, student, employee, president, wife, etc. For instance, student is an anti-rigid type, i.e., a type that classifies its instances contingently. In other words, no student is necessarily a student (in a modal sense) and every instance of student can cease to be a student without ceasing to exist. Moreover, in order to be student, someone must be enrolled in an educational institution, i.e., being a student is a type of relationally dependent property.

The contributions of this chapter are three-fold. Firstly, we contribute to the identification of three new Ontological Anti-Patterns for Conceptual Modeling, in general, and for OntoUML, in particular. We do that by carrying out an empirical qualitative approach over a model benchmark of 54 OntoUML models. Secondly, after precisely characterizing these anti-patterns, we propose a set of refactoring plans that can be applied to the models in order to eliminate the possible unintended consequences induced by the presence of each of these anti-patterns. Finally, we present an extension for Menthor Editor, an OntoUML model-based editor with a number of features for: (i) automatically detecting anti-patterns in user models; (ii) 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 characterizes a modeling error, either because it allows unintended model instances or because it forbids intended ones; (iii) automatically executing refactoring plans to exclude these unintended model instances, which can take the form of Object Constraint Language (OCL) [18] constraints or direct interventions in the model.

The remainder of this chapter is organized as follows: in Section 1.2, we briefly elaborate on the modeling language OntoUML and some of its underlying ontological categories; Section 1.3 discusses the notion of Ontological Anti-Patterns as taken in this work with some brief discussion of related work; Section 1.4 discusses the anti-pattern elicitation method employed here and characterizes the model benchmark used in this research; Section 1.5 presents the elicited Ontological Anti-Patterns with their unintended consequences, as well as possible solutions for their rectification in terms of model refactoring plans; Section 1.6 discusses empirical data about the occurrence of these anti-patterns in our model repository and presents the results of an additional industrial empirical study aimed at investigating the accuracy of our anti-pattern catalog as well as the efficacy of the

proposed refactoring plans; Section 1.7 elaborates on the extensions implemented in the OntoUML editor taking into account these anti-patterns; Finally, Section 1.8 presents some final considerations.

1.2. Ontological Foundations

OntoUML is a language for Ontology-driven Conceptual Modeling, whose meta-model was designed to comply with the Unified Foundational Ontology (UFO) [11]. OntoUML has constructs that represent the ontological distinctions put forth by UFO as well as constraints on how these constructs can be combined. These OntoUML metamodel constraints are derived from the UFO axiomatization of the respective ontological distinctions. In the sequel, we briefly introduce some few OntoUML constructs that are germane for the purposes of this article. For a complete presentation and formal characterization of the language, the reader is referred to [11].

One of UFO's fundamental notions is that of rigidity. Rigidity is a property of types (i.e., a meta-property) that specifies whether its instantiation is essential or accidental for its instances. Rigid types are the ones that aggregate essential properties to all its instances, requiring them to instantiate it while existing (e.g. Car, Forest, Person, Band). Anti-rigid types, in contrast, aggregate accidental properties for its instances. If an individual instantiates an anti-rigid type in a given situation, there is at least one other alternative situation where it does not do so (e.g. Adult, Student, Customer, Husband).

In this paper, for the sake of conciseness in our presentation, we restrict ourselves to a fragment of OntoUML focused on sortal types. A sortal type is a type that carries a uniform principle of identity for its instances. Rigid sortal types, as the name suggests, are rigid types whose instances follow a unique principle of identity (e.g. person, man, dog). A particular type of rigid sortal is the substance sortal type, which supplies a principle of identity for its instances (e.g. person, organization, car). In UFO and, hence, in OntoUML, substances sortals are either kinds, collectives or quantities. Since the examples used in this chapter are centered around functional entities of everyday experiences (e.g., people, organizations, computer systems), focusing on the notion of kind will suffice and will not cause a loss of generality. Still within the category of rigid sortals, we have rigid types that specialize substance sortals. These are termed *subkinds* (e.g. man, woman, Brazilian). Among the category of anti-rigid sortals, we have phases (e.g. adult, puppy, living) and roles (e.g. employee, student). In summary, in OntoUML, a sortal type is either a substance sortal (e.g., a kind) or a subtype of unique substance sortal (e.g. a subkind, role or phase). For more information, please see [11].

For this chapter, the notion of «role» is central. A Role is a type that is not only anti-rigid but also relationally dependent, i.e., a type whose instantiation depends on individuals bearing relational properties of certain sorts. In other words, in order to instantiate (play) a certain role, an individual must participate in certain material relations. Examples of roles and their respective relational dependencies include: a student and its enrollment in a school, an employee and its employment contract with a company, and a father with his legal children.

UFO (and, hence, OntoUML) distinguishes between two broad categories of relations, namely formal and material relations. In order for a set of individuals to participate in a material relationship (i.e., a relation instance), they must be connected (mediated) by yet another entity termed a relator. A relator is said to be the truthmaker of that material relationship [10]. Take for instance the relation *studies-at* that can hold between a student and an university. For it to be true that Camila studies at University of Trento, there must be a particular enrollment connecting them (the relator). Other examples of material relations and their respective relator types are *married-to* and *marriage*, *works-at* and *employment*, and *being-the-president-of* and *presidential mandate*.

In OntoUML, we represent material relations using the «material» stereotype, and relator types are marked with the «relator» stereotype. Relations stereotyped as «mediation» are used to connect the related types (typically «role» classes) to the corresponding relator type (stereotyped as «relator»). Finally, the «derivation» relation is a formal relation connecting a material relation to the relator type whose instances are the truthmakers of the instances of that material relation (e.g., it connects the *married-with* relation to the relator type *Marriage* such that, for instance, *married-with*(Tiago,Camila) holds iff there is an instance of *Marriage* mediating the two).

Relators are examples of existentially dependent entities in UFO and OntoUML. In fact, they are examples of multiply existentially dependent entities, i.e., entities that depend on a multitude of individuals [11]. They belong, however, to a broader class of existentially dependent entities termed moments (also termed tropes, abstract particulars, objectified properties or particularized properties) [11]. So, relators are multiply dependent moments. In UFO; a moments that is existentially dependent on single entity is either a mode (e.g., a symptom, a belief, a disposition) or a quality (e.g., the color of an apple, the weight of a person) [11].

1.3. What is an Anti-pattern in Ontology-driven Conceptual Modeling?

Inspired by the Gang of Four’s seminal work on design patterns [9], Andrew Koenig coined the term “anti-pattern” [15]. His original definition states that an anti-pattern is just like a pattern, in the sense that it is a recurrent solution for a given problem, but it is one that entails more bad consequences than good ones. To an anti-pattern, a proper pattern is to be associated as a better solution to the problem at hand. “The Blob” is an example of an anti-pattern for object-oriented software design [5]. The authors of that anti-pattern describe it as a procedural style design in an object-oriented environment, which cause one or few classes to aggregate most of the functionalities of the system, while the remainder classes just carry basic data or perform simple tasks.

In the late 90’s, Fowler introduced the notion of *code smells* (or bad smells) [8]. The concept of code smell stands for a distinct code structure that requires careful attention because it is likely to produce maintainability and comprehensibility issues on the software being developed. The name *smell* conveys the idea of something “fishy” regarding the code. In their proposal, code smells are guides

for code refactoring. The most basic example of a code smell presented by Fowler is the “Duplicate Code”. Its definition is straightforward, a given code structure repeated in different places throughout a larger body of code.

In our work, we bring the ideas of anti-patterns and code smells to ontology-driven conceptual modeling, in general, and to OntoUML, in particular. Instead of software development, we deal with the problem of accurately capturing and formalizing a given conceptualization into a domain conceptual model. Instead of dealing with code structures, we deal with model structures. The idea is that “fishy model structures”, in this in this context, lead to domain misrepresentations from an ontological point of view.

We define an ontological anti-pattern as a modeling pattern that, despite producing syntactically valid conceptual models, it is prone to be the source of domain-related ontological misrepresentations. An anti-pattern must have a defined structure and refactoring options (or rectification plans) associated to it. Note that our definition combines the ideas behind anti-patterns and code smells. On one hand, we identify structures that capture recurrent modeling decisions that are prone to decrease model quality. Moreover, we combine these structures with appropriate solutions. In this spirit, our anti-pattern definition resembles the original one of Koenig. On the other hand, because our anti-patterns point to decisions that are not always incorrect and mean to serve as a guide for modelers to validate (and possibly refactor) their models, they also resemble code smells. We emphasize that our notion of anti-patterns is not as synonym for a modeling error or a bad modeling practice. One should think of it as model fragments that are worth “putting under the microscope”. On one hand, unlike Koenig’s definition of anti-pattern, the modeling solution at hand is not necessarily a bad one. On the other hand, unlike code smells, the decisions do not imply maintainability or architectural issues.

We should also differentiate an anti-pattern from a recurrent modeling expression that when created in a particular language always represents an error. Take, for instance, the situation in which an anti-rigid type is represented as a supertype of a rigid type, i.e., a type that classify its instances necessarily (e.g., the natural kind Person). This situation always leads to a logical contradiction [11]. As such, this is not an example of an anti-pattern, it is a logical (and ontological) mistake. In order to avoid this mistake, a language should include a syntactical constraint in its metamodel to always deem such a situation as syntactically invalid (i.e., as a syntactical error). This is exactly what is done in OntoUML [11], in which the incorporation of ontological constraints in its meta-model prevents the representation of ontologically non-admissible states of affairs [11], i.e., representations that would contradict UFO’s *domain-independent* axiomatization. However, as discussed in [11] there is no way a domain-independent language can exclude in *a prioristic* manner models that admit as instances representations that contradict *domain-specific axiomatizations*, i.e., instances that represent unintended state of affairs according to domain-specific conceptualizations [11].

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. Suppose that this model has as a possible instance one that represents a state of affairs in which

the Donor, the Donee and the Transplant Surgeon are one and the same Person. Now, suppose that this instance represents an unintended state of affairs according to our assumed conceptualization of the domain of transplants. Notice that the state of affairs represented by this instance is only considered inadmissible (unintended) due to domain-specific knowledge of social and natural laws. As discussed in [11], the presence of this unintended instance is typically caused in OntoUML models by the introduction of a particular model fragment. Such a model fragment would be an example of what we term here an Ontological Anti-Pattern.

1.3.1. Related Works: Anti-Patterns in the Semantic Web

Since Koenig’s original proposal [15], the concept of anti-pattern has been applied in a variety of fields other than software design. We are unaware of other works in the literature that systematically study anti-patterns for ontology-driven conceptual modeling. For this reason, we compare our take on anti-patterns to related in the context of the semantic web. As it shall become clear in the sequel, when referring to anti-patterns or similar notions, these works assume an interpretation of the term that differs significantly from the one employed here.

In [20], Roussey et al. focus on OWL anti-patterns. For them, an OWL anti-pattern is a pattern that is commonly used by domain experts in their OWL implementations and that normally result in inconsistencies. They argue that anti-patterns come from a misuse and misunderstanding of description logics expressions by ontology developers. Their anti-patterns are classified in three exclusive categories: (i) logical, which represents errors that reasoners detect; (ii) cognitive, which classifies possible modelling errors that are not detected by reasoners; and (iii) style (originally named “Guidelines”), which describes logically correct expressions made unnecessarily complex, e.g using the complement operator instead of disjointness. Our approach differs from theirs in the nature of the problems we investigate. First, we are not concerned with inconsistency issues here because the most typical logical mistakes that are of an ontological nature are prevented by OntoUML’s metamodel by design. Moreover, we are also not concerned with anti-patterns as conventional guidelines in the same spirit because we want to improve model precision and accuracy, not readability or maintainability. Lastly, cognitive anti-patterns include problems like expression redundancy, which is also out of the scope of our anti-pattern validation approach.

Poveda-Villalón et al. build their validation approach [19] around the concept of common ontology pitfalls. A pitfall stands for an anomaly or worst practice in ontology design identified through empirical analysis. As Roussey et al., they also see these recurrent problems as misunderstanding and misusing the description logics constructs. The authors classify the common pitfalls identified in RDF and OWL ontologies, from two different perspectives. First, regarding the type of problems they can cause, the pitfalls are classified into structural, functional and usability pitfalls; second, regarding the quality criteria they affect, they are classified into the categories of consistency, completeness and conciseness pitfalls. The use of pitfalls for ontology validation is supported by a tool named OOPS! Ontology Pitfall Scanner [19]. In our work, we require anti-pattern definitions

to prescribe finite and identifiable model structures. In contrast, many pitfalls proposed in OOPS! do not have this property. An example is the pitfall “Merging different concepts in the same class”, which rely on an analysis of class names, to identify the use of the operators “and” or “or” on the class name.

Baumeister and Seipel investigate recurrent anomalies in OWL ontologies enriched with SWRL rules [4]. They distinguish between four types of anomalies, namely circularity, inconsistency, redundancy, and deficiency. Circularity refers to circular definitions in the ontology, involving either classes, properties and rules, that hinder the performance of reasoners. The inconsistency category describes anomalies that produce actual logical contradictions in the ontology. A very common example, according to the authors, is the definition of a class as subclass of two disjoint classes. Anomalies related to redundancy point out knowledge that can be excluded from the ontology without changing its semantics. Lastly, deficiency anomalies refer to problem of completeness, understandability and maintainability of the ontology. An example of deficiency is the “lazy class”, which stands for classes that have little use in practical applications of the ontology.

In summary, our approach differs from all the aforementioned efforts on a very key aspect. We do not intend that our anti-patterns support modelers in building the models correctly, but in building the (ontologically) correct model for the domain. In other words, our focus is on model validation instead of model verification [11]. Because of that, we do not focus on modeling patterns that are always wrong (e.g. patterns that lead to logical inconsistency). Instead, we focus on anti-patterns that represent structures that frequently (as empirically demonstrated) cause dissociations between the set of valid instances admitted by a model and the set of intended instances admitted by the underlying domain conceptualization [11].

1.4. Empirically Uncovering Ontological Anti-patterns

1.4.1. Method

Our approach to identify ontological anti-patterns is an empirical qualitative analysis. It starts with the selection of a particular model for analysis. Within the selected model, the second step is to select relevant fragments for analysis. Such fragments can consist of a whole diagram, a subset of a diagram or even a new “artificial” diagram produced for the sake of analysis (model inspection).

Step three is to inspect the selected portion of the model and identify possible problems. We conduct this activity using visual model simulation [12]. This simulation consists in converting OntoUML models into Alloy [14] specifications, generating possible model instances and contrasting these instances with the set of intended instances of the model. The set of intended instances correspond to those that represent intended state of affairs [11] according to the creators of the models. Upon the identification of a mismatch, we register it as a potential problem.

After detecting a possible problem, we analyze the model in order to identify which structures (i.e., combination of language constructs) caused that problem.

In the sequence, we interact with the modelers (when available) or inspect the documentation accompanying the model to define whether the identified structure is indeed problematic. If it is, we propose a possible solution to rectify the model and register it as a (problem, solution) pair. With a recently modified model, we go back to step three. This iteration is repeated until no more problems can be identified in the fragment and then, another fragment is selected. The analysis stops whenever we inspect all relevant model fragments. After inspecting each model, we analyze the generated problem-solution pairs in order to generalize them into pairs of anti-patterns and refactoring plans.

1.4.2. Model Repository

Our empirical analysis for uncovering anti-patterns was performed using a repository of 54 models¹. Tables 1.1 and 1.2 provide an summary on the models our repository according to the following dimensions:

- **Name:** the name provided by the model’s authors. If none was provided, we baptized the model with a chosen intuitive name.
- **Context:** the scenario in which the model was developed. The model is placed in one of the following categories: Government Project, for models developed by or in cooperation with governmental entities; Industrial Project, for models produced by or in cooperation with private companies; Graduate course assignment (assignment) for models produced by graduate students as a final assignments of an “Ontology Engineering” 60-hour course offered by the Graduate School on Informatics of the Federal University of Espírito Santo; PhD Thesis (PhD), MSc Dissertation (MSc) and BSc Monograph (Bsc), for models produced as outcome of academic research in each of these respective levels; and Other, for the models that do not fit the aforementioned categories.
- **Purpose:** describes the motivation for building the model at hand. The category termed interoperability is intended for those models created as reference models for semantic interoperability between agents and/or systems; ontological analysis (ontol. analysis) classifies models developed to evaluate conceptual modeling languages or other domain formalizations; reference model (reference model) classify those models that are proposed as reference ontologies for a given community; knowledge-based application (kb application) stands for models created to support the development of this kind of applications; lastly, unspecified categorizes models whose authors did not provide further characterizing information on this topic.
- **Expertise (Exp.):** describes the authors’ familiarity to the OntoUML language during the development of the ontology. We classify a modeler as a beginner (beg) if he/she had at most a year of experience using the language, and as advanced (adv) otherwise.
- **Number of Participants (Part.):** provides an estimated number of people involved in the production of the ontology. We consider as participants

¹The conceptual models in this table are publicly available at: <http://www.menthor.net/model-repository.html>. The few exceptions of missing models are due to non-disclosure agreements that prohibited their publication.

those actively involved in either scope definition, knowledge acquisition, design or evaluation. For published models, we estimate the number of participants using the authors of the publication, whilst for those models developed in the context of theses, we consider the student and all official supervisors.

Table 1.1 lists the 32 models developed as graduate course assignments, whilst Table 1.2 present the remainder 22. From these, 11 models were developed during academic research without industry collaboration. An example is The Configuration Management Task Ontology [6], a product of a Masters dissertation. Furthermore, seven models had total or partial participation of private companies and/or governmental organizations. The most significant one is the MGIC Ontology [3], developed in the context of a research project with a regulatory agency responsible for controlling ground transportation services in Brazil. Finally, the development contexts of 4 models were not available.

Concerning the purpose for which the models have been created, the repository contains 10 models (16%) that are intended to serve as a reference domain or core ontologies (e.g. UFO-S [17] for the domain of services). Another 10 models (16%) have been developed in order to perform ontological analysis on existing formalizations, databases or modeling languages. An example is the refactoring of the Conceptual Schema of Human Genome presented in [16]. The repository also contains 8 models (13%) designed for supporting the development of knowledge-based applications, 6 (10%) whose main intention was to support semantic interoperability between systems and/or organizations, and only two (3%) for the purpose of enterprise modeling. For the remainder 26 models (42%), there is no information w.r.t. this classification dimension.

Regarding the modeler's overall expertise in OntoUML, 22 models (41%) have been developed by beginners (18 of these models are also graduate course assignments) and 32 (59%) developed by experienced modelers. Finally, we look into the total number of modelers involved in the model construction. A single person participated in the development most of the models (35 models, roughly 65%). However, 15 models were the product of collaboration efforts between 2-4 people and 4 models involved 7-10 conceptual modelers.

1.5. Ontological Antipatterns

In this section, we extend the anti-pattern library presented in [13, 22, 23] with three anti-patterns centered around OntoUML's «role» construct. In order to facilitate learning and usage of these anti-patterns, we describe them following a consistent format. First, we discuss each anti-pattern in natural language. Next, for each anti-pattern, we present a table that summarizes the anti-patterns essential characteristics. Lastly, we exemplify that anti-pattern with one its occurrences in the models presented in the model repository considered in this study. For each anti-pattern, we then describe the following information:

- **Name and Acronym:** unique identification in the catalogue;
- **Description:** a general description of the anti-pattern;

Table 1.1. Overview of models in the repository developed as assignments.

Model	Context	Main Purpose	Exp.	Part.
Gi2MO Refactored	assignment	ontol. analysis	adv	1
Internal Affairs Ontology	assignment	ontol. analysis	adv	3
Library Model	assignment	unspecified	beg	1
Public Tenders Model	assignment	unspecified	adv	1
Social Contract Model	assignment	unspecified	beg	1
Clergy Model	assignment	unspecified	beg	1
FIFA Football Model	assignment	unspecified	adv	1
IDAF Model	assignment	unspecified	adv	1
University Model	assignment	unspecified	beg	1
GRU MPS.BR Model	assignment	unspecified	beg	1
Experiment Model	assignment	unspecified	beg	1
Parking Lot System	assignment	unspecified	adv	1
Quality Assurance Ontology	assignment	unspecified	adv	1
Music Ontology Refactored	assignment	ontol. analysis	adv	1
Internship Model	assignment	unspecified	adv	1
G.809 Model	assignment	unspecified	beg	1
Online Mentoring Model	assignment	unspecified	adv	1
Help Desk System Model	assignment	unspecified	beg	1
IT Infrastructure Model	assignment	unspecified	beg	1
Photography Model	assignment	unspecified	beg	1
FIRA Ontology Refactored	assignment	onto analysis	adv	1
Banking Model	assignment	unspecified	adv	1
Chartered Service Model	assignment	unspecified	adv	1
Health Organization Model	assignment	unspecified	beg	1
Bank Model 2	assignment	unspecified	adv	1
PROV Ontology Refactored	assignment	ontol. analysis	beg	1
WSMO Refactored	assignment	ontol. analysis	beg	1
Recommendation Model	assignment	kb application	beg	1
Project Management Model	assignment	unspecified	beg	1
Construction Model	assignment	unspecified	beg	1
Stock Model	assignment	unspecified	beg	1
Real State Model	assignment	unspecified	beg	1

Table 1.2. Overview of models in the repository developed in various contexts.

Model	Context	Main Purpose	Exp.	Part.
OpenFlow Ontology	BSc	kb application	beg	2
CMTO	MSc	interoperability	adv	2
OntoEmerge	MSc	kb application	adv	8
PAS 77:2006 Ontology	MSc	ontol. analysis	adv	4
Cloud Vulnerability Ontology	MSc	reference model	adv	4
OntoUML Org Ontology	MSc	ontol. analysis	adv	2
ECG Ontology	MSc	interoperability	adv	3
Requirements Ontology	MSc	kb application	adv	2
Open proVenance Ontology	PhD	reference model	adv	3
CSHG Refactored	PhD	ontol. analysis	adv	3
UFO-S	PhD	reference model	adv	7
MGIC Ontology	government	enterprise model	adv	10
MPOG Ontology Draft	government	reference model	beg	7
OntoBio	government	interoperability	adv	3
Normative Acts Ontology	government	reference	adv	3
G.805 Ontology	industry	ontol. analysis	adv	4
G.805 Ontology 2.0	industry	kb application	adv	3
G.800 Ontology	industry	kb application	adv	3
TM Forum Model	other	unspecified	beg	1
School Transport Model	other	application	beg	1
ERP System Model	other	interoperability	adv	1
Inventory System	other	interoperability	adv	1

- **Structure:** structural definition of an anti-pattern, consisting of (i) *pattern roles*, to describe the elements that participate in the anti-pattern, their possible stereotypes and cardinalities; and (ii) a *generic example*, i.e., a graphical exemplification of the anti-pattern’s generic structure;
- **Refactoring Plan:** recurrent alternative solutions defined as a sequence of actions that can be used to rectify the model containing the problem created by the occurrence of that anti-pattern. Includes: (i) *constraint definition*, to indicate the specification of an OCL constraint; (ii) *element modification*, to indicate a change in a model element (e.g. meta-property change); and (iii) *element creation*, to indicate the introduction new elements to the model.

Note that mediation (and characterization) relations always have a minimum cardinality of at least 1 and an immutability meta-property (i.e., *readOnly = true*) in their target association end, i.e., on the association end of this relation connected to the mediated (characterized) type. For the sake of brevity, we refrain from repeating these constraints in each depiction of these relations when presenting anti-pattern structures.

1.5.1. Relator Mediating Rigid Types

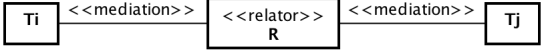
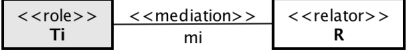
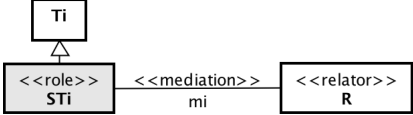
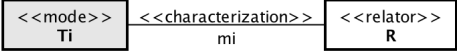
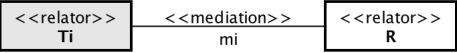
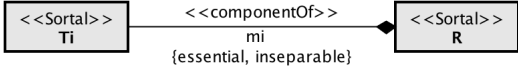
The first anti-pattern we introduce here is called *Relator Mediating Rigid Types* (*RelRig*). Its occurrence is characterized by a «relator» class R connected to one or more rigid sortal classes $T_1..T_n$ (e.g. «kind», «subkind») through «mediation» relations. *RelRig*’s main characteristics are presented in Table 1.3.

A type T_i is deemed relationally dependent if it is connected to a «relator» R through a «mediation» m . In other words, for an individual x to be an instance of type T_i there must exist an instance of R that mediates x [11]. For instance, *Student* is typically a relationally dependent type whose instances are mediated by instances of *Enrollments*, i.e., in order for someone to be a *Student*, she must be mediated by a particular *Enrollment*. Notice that in this example, *Student* is a «role». The issue at stake with this anti-pattern is that rigid sortal types aggregate essential properties of individuals and thus are usually relationally independent types [11]. Mediation relations, in contrast, are commonly used to define roles, i.e., anti-rigid types that aggregate accidental relational properties.

In our empirical investigation, we confronted ontologists regarding the models in which they judged that this anti-pattern really introduced an error (i.e., an unintended representation). Our goal was to find out which was the original modeling intention that caused (by mistake) the introduction of this anti-pattern in the model.

- (i) the mediated type T_i at hand should actually be modeled as a role. For instance, suppose one represents a type Public Enterprise as a «subkind» of Enterprise that is connected to a «relator» type Public Offering. In this case, if an Enterprise can exist before going public, it means that being a Public Enterprise is a contingent (and relationally dependent) subtype of Enterprise;

Table 1.3. Summary of the *RelRig* anti-pattern

Name (Acronym)		Description
Relator	Mediating Rigid Types (RelRig)	A relator connected to one or more rigid types through mediations. This is a potential problem because relational dependencies are commonly defined for anti-rigid types.
Pattern Roles		
Mult.	Name	Allowed Metaclasses
1	R	«relator»
1..*	m_i	«mediation»
1..*	T_i	«kind», «quantity», «collective», «subkind»
Generic Example		
		
Refactoring Plans		
1. Role: Change T_i 's meta-class to role.		
		
2. Role subtype: Create a role that specializes T_i and reconnect the respective m_i to it.		
		
3. Reversed dependency: a) change T_i to mode and M_i to characterization or b) change T_i to relator and invert m_i .		
<p>(a) </p> <p>(b) </p>		
4. Bidirectional dependency: transform R into a sortal and change m_i into an <i>essential</i> and <i>inseparable</i> parthood relation.		
		

- (ii) the «mediation» relation was not supposed to be declared with a mandatory cardinality constraint from the perspective of T_i (i.e., in the association end opposite to T_i in the «mediation» relation). Now, if this cardinality constraint is an optional one, this means that there is a subtype ST_i of T_i that has been omitted in the model. Notice that ST_i is defined as the type instantiated by the instances of T_i when mediated by instances of the «relator» R at hand. As such, ST_i is by definition a «role» since: it is anti-rigid - the instances of ST_i can exist (as instances of T_i) without being instances of ST_i ; instances of T_i become instances of ST_i when mediated by instances of R . As an example of this situation suppose that someone represents a constraints stating that instances of *Person* are necessarily connected to a *Citizenship* (a «relator»). After inspection, the modeler can realize that actually there are people that do not have a *Citizenship* (e.g., people that were born but not yet formally registered) and that there is a subtype of *Person* (namely *Citizens*) that have a *Citizenship* connecting them to a *Country*;
- (iii) there is indeed an existential dependency that should be represented in the model but in the reverse direction, i.e., it is the type T_i that has instances that are existentially dependent on corresponding relators, not the other way around. This means that this otherwise mediated type is not a sortal type but a type whose instances are moments. This case has still two variants:
 - (a) the case in which T_i is a type whose instances are modes, in which case the «mediation» relation between T_i and R should be replaced by a «characterization» relation. As an example, take the case in which we have a *Bundle of Rights* that one has in the scope of a particular *Employment*;
 - (b) the case in which T_i is a type whose instances are relators, in which case the mediation relation between T_i and R is indeed a mediation relation but it is T_i that connects instances of R to instances of another type and not the other way around. As an example, suppose we have a *Chairmanship* as a *Parliament Commission Chairman*, which existentially depends on a *Mandate* «relator» (as well as on a specific *Commission*). As this example shows, finding out that *Chairmanship* is actually a relator triggers a process of instantiating the entire Relator Pattern [12], which in turn leads to the possible discovery of the role *Parliament Commission Chairman* (a «role» played by instances of the «role» *Congressman*) as well as the type *Commission*;
- (iv) finally, we have a case in which there are indeed mutual existential dependence relationships between instances of T_i and instances of R . However, as discussed in depth in [11], sortals are not existentially dependent on entities from which they are mereologically disjoint. In other words, if a sortal x is existentially dependent on sortal y then either x must necessarily be a part of y (which we term an inseparable part) or y must necessarily be a part of x (which we term an essential part) (see [11] for details). As an example of this case, suppose that one wants to model that a *Social Contract* R creates

a *Joint Venture* between a number of organizations $T_1..T_n$ that only exist for the scope of that transient *Joint Venture*, i.e., once the social contract ceases to exist then the organizations themselves cease to exist. In this case, the relation between an *Organization* instance of T_i and an instance r of R is not one of mediation but one of inseparable and essential parthood. In other words, r is defined as a mereological sum of instances of T_i and each instance of T_i only exist as part of that *Joint Venture* r .

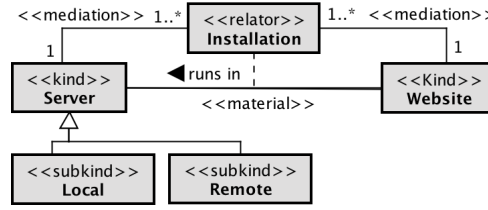


Figure 1.1. *RelRig* example extracted from the PAS-77 Ontology [7].

Figure 1.1 presents a model fragment adapted from the PAS-77 Ontology [7] that contains a *RelRig* occurrence. This simplified fragment states that a *Website* has to be installed in one or more *Servers*. It also states that a *Server* has at least one *Website* running on it. Finally, the authors distinguish *Servers* with regard to where they are located, i.e., within a company or in a remote location.

As it is, the model does not account for recently acquired or formatted computers, which would not have any application running on them. If this is a desired possibility, the authors have improperly characterized *Server* as a «kind». Instead, it is a «role» a *Computer System* play when hosting an application or a website. Regarding the representation of *Website*, the model forbids the existence of instances of website that are not installed in any server. If we consider that a *Website* exists since its development, when it is not running anywhere, then we may want to include a new «role» in the model, namely *Active Website*.

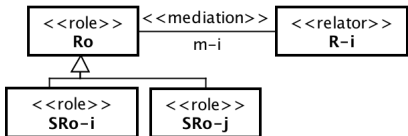
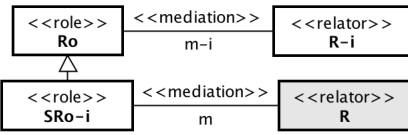
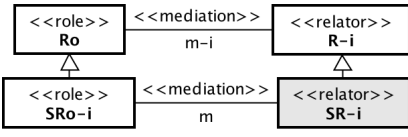
1.5.2. Free-Floating Role Specialization

The *Free-Floating Role Specialization (FreeRole)* anti-pattern focuses on uncovering (possibly) missing conditions for certain roles to be instantiated. Its generic structure consists of: a «role» Ro connected to relators $R_i..R_n$ through mediations $m_i..m_n$; the «role» Ro is specialized by one or more roles $SRo_i..SRo_n$; the specializing $SRo_i..SRo_n$ are not directly connected to any additional «mediation» relations. The essential features of the *FreeRole* anti-pattern are summarized in Table 1.4.

Our empirical investigation suggests that *FreeRole* occurrences are often modelling errors, which can be corrected by applying one of the following role specialization refactoring plans:

- (i) *Derived sub-role*: this should be applied when SRo_i is instantiated according to a derivation rule. For example, a *Person* plays the role of *Student* when

Table 1.4. Summary of the *FreeRole* anti-pattern.

Name (Acronym)		Description
Free Role Specialization (FreeRole)		A role connected to a relator through a mediation, is specialized by one or more roles, which, in turn, are not connected to additional mediation relations. This pattern suggests that a properly characterized instantiation criteria for these specializing roles might be missing.
Pattern Roles		
Mult.	Name	Allowed Metaclasses
1	Ro	«role»
1..*	m_i	«mediation»
1..*	R_i	«relator»
1..*	SRo_i	«role»
Generic Example		
 <pre> classDiagram class Ro["<<role>>\nRo"] class Ri["<<relator>>\nR-i"] class SRo_i["<<role>>\nSRo-i"] class SRo_j["<<role>>\nSRo-j"] Ro --> Ri : m-i Ro < -- SRo_i Ro < -- SRo_j </pre>		
Refactoring Plans		
<p>1. Derived sub-role: add proper OCL constraint defining the derivation rule at hand.</p> <p>2. Role-of-a-role: create a new relator R and connect it to SRo_i using a new mediation relation.</p>		
 <pre> classDiagram class Ro["<<role>>\nRo"] class Ri["<<relator>>\nR-i"] class SRo_i["<<role>>\nSRo-i"] class R["<<relator>>\nR"] Ro --> Ri : m-i SRo_i --> R : m </pre>		
<p>3. Intentional sub-role: create a new relator SR_i by specializing R_i and connect it to SRo_i using a new mediation.</p>		
 <pre> classDiagram class Ro["<<role>>\nRo"] class Ri["<<relator>>\nR-i"] class SRo_i["<<role>>\nSRo-i"] class SR_i["<<relator>>\nSR-i"] Ro --> Ri : m-i SRo_i --> SR_i : m Ri < -- SR_i </pre>		

she enrolls at an *Educational Institution*; *Students* are classified as *Freshmen*, if enrolled for less than a year, and as *Seniors* otherwise;

- (ii) *Role-of-a-role*: this should be applied when SRo_i is instantiated due to an additional relational dependency. To exemplify, we use the *Student* concept once more. A *Student* becomes an *Intern* when she starts an *Internship* program (a new relator) within a *Company*;
- (iii) *Intentional sub-role*: this is suggested when SRo_i is instantiated due to a particular subtype of the generic dependence relation characterizing role *Ro*. As an example, consider the concepts of *Bachelor Student* and *Graduate Student*. These are both types of *Student*, and thus, both are defined by *Enrollments*. However, each of these student roles requires particular subtypes of *Enrollments* with particular characteristics. For instance, unlike a *Bachelor Enrollment*, a *Graduate Enrollment* requires the assignment of a *Thesis Supervisor*.

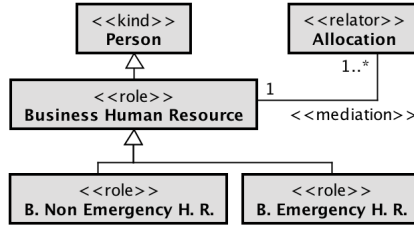


Figure 1.2. *FreeRole* example extracted from the OntoEmerge Ontology.

The small model fragment depicted in the right side of Figure 1.2 corresponds to a *FreeRole* occurrence encountered in OntoEmerge [2], an ontology for the emergency domain. In this current presentation, the model accepts that for a given allocation, a person can play an emergency-related role at a given time and a non-emergency one at another, without any apparent property being changed in the allocation itself. The intentional sub-role pattern can then be used to provide a more suitable representation for this domain. In this case, two subtypes of the relator would be created, one for each type of business resource role.

1.5.3. Multiple Relational Dependency

Roles are relationally dependent types. This means that an individual plays (or instantiates) a role whenever a particular type of material relation is established with another individual. OntoUML, however, does not impose any constraint with respect to the number of material relations that can be used to characterize a role, each possibly representing a different source of relational dependence. The *Multiple Relational Dependency* (*MultDep*) anti-pattern focuses the attention of the modeler on types whose definition includes multiple material relations characterizing a role. Structurally, its identification consists of a sortal class directly

connected to two or more «mediation» relations. Table 1.5 provides an overview of the *MultDep* anti-pattern.

A *MultDep* occurrence requires attention because, through our empirical analysis, we identified that it might indicate:

- (i) The model might be over-constrained, in the sense that one or more «mediation» relations represented as mandatory relations are in fact optional relations. This could in itself can potentially hide other implicit role types;
- (ii) In addition to (i), there is a particular order in which two or more optional «mediation» relations can be established;
- (iii) The model is lacking a relationship between two or more of the «relator» types that characterize the multiple sources of relational dependence.

The first step to analyze a *MultDep* occurrence is to verify whether the ascribed mediation relations are indeed mandatory. If that is not the case for a given mediation relation, the appropriate refactoring solution is the transformation of that mediation relation into an optional one. This is followed by the inclusion of a new role in the model, which would “carry” the mediation relation for itself. In the particular case in which a modeler concludes that two or more mediation relations are optional, an evaluation of whether they are established in a particular sequence is in order. To exemplify, consider two roles that can be played by a person: the student role, when related a school (via an enrollment relator), and the employee role, when related to a company (via an employment relator). A priori, one cannot make any assertion w.r.t the order in which one becomes a student and an employee. Conversely, if we consider the roles of student and intern (and their implied dependencies) there is clearly a restriction regarding the order in which these two roles should be played, namely, that a person could only become an intern if she is already a student. This means that the internship (a relator) connecting an intern to a company depends on the existence of an enrollment (another relator) connecting that intern as a student to a school.

If the class that characterizes the *MultDep* occurrence has more than two mediations connected directly to it, a modeler might refactor it by simultaneously defining ordered and unordered optional mediation relations. Nonetheless, when creating unordered dependencies, all new roles should specialize the same super-type. If an order is required, the modeler should create the new roles as subtypes of one another, i.e., they should obey a pre-defined hierarchical order.

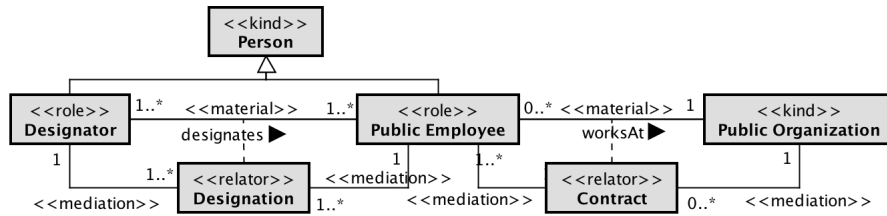
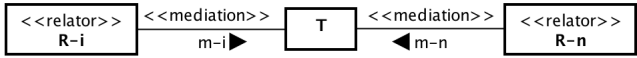
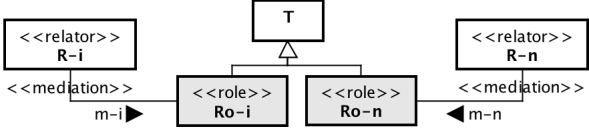
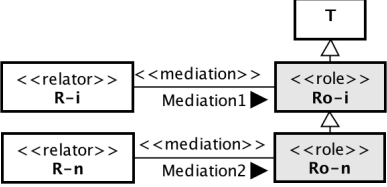
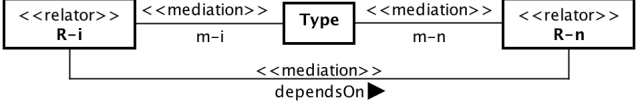


Figure 1.3. *MultDep* example extracted from the MGIC Ontology [3].

Table 1.5. Summary of the *MultiDep* anti-pattern.

Name (Acronym)		Description
Multiple Relational Dependency (MultiDep)		A sortal class directly connected to two distinct relators through mediation relations. This structure might indicate redundancy, over constraining or scope issues.
Pattern Roles		
Mult.	Name	Allowed Metaclasses
1	T	All sortal types
2..*	m_i	«mediation»
2..*	R_i	«relator»
Generic Example		
		
Refactoring Plans		
<p>1. Unordered optional dependency: For each selected m_i, create a direct subtype of T and connect m_i to it.</p> 		
<p>2. Ordered optional dependency: For each selected m_i, create a new role following the chosen instantiation order.</p> 		
<p>3. Relator dependency: For each identified dependency, create a mediation connecting the two relators.</p> 		

Regardless if a *MultDep* occurrence identifies optional mediation relations or not, the next step is to analyze if there are relational dependence sources that depend on other relational dependence sources. In other words, if there are existential dependence relations between the relators that embody these different relational dependence sources. To illustrate this point, consider the following scenario: *A person becomes an undergraduate student when she enrolls in a bachelor program at a university, e.g. Computer Science or Philosophy. A unique number identifies each enrollment. Victor, a very curious and dedicated young man, decides to pursue, simultaneously, a major in Philosophy and Computer Science. To do that, he would need to enroll two times at the university. After his enrollments, Victor wants to apply for Logics 101 as a Computer Science major and apply for Sociology 101 as a Philosophy major. To do that, each course application must not only identify Victor as the applying student, but also identify the particular enrollment he is using to apply.*

The identification of the enrollment in the course application suggests that a course application is existentially dependent of the student, the course and the actual specific enrollment associated to a particular major. We propose the formalization of this dependency in an OntoUML model as a new «mediation» relation from course application to enrollment in a major.

The *MultDep* example we selected comes from the ontology developed in the MGIC project, whose goal was to develop a model for knowledge and information management for the Brazilian Ground Transportation Regulatory Agency (ANTT). The small model fragment depicted on Figure 1.3 states that a public employee has one or more designations (for a permanent position in the public administration) and exactly one valid contract. The sensitive issue here is that, according to the law governing these public positions in Brazil, a given contract is grounded on a particular designation (a part of the public employment process in Brazil). As presented in Figure 1.3, the model allows for an independent variation of contracts and designations, which should not be allowed. Refactoring plan 3 should be adopted to rectify this model, thus formalizing an existential dependency from contract to designation.

1.6. Anti-Pattern Evaluation

We define semantic anti-patterns as being error-prone recurrent modeling decisions. Therefore, in order to evaluate them, we analyze two dimensions. This first is frequency, which measures how recurrent the modeling decisions are. The second is usefulness, which indicates how error-prone the anti-patterns are. In this section, we report on the results of two studies we conducted to assess each of these aspects.

For the sake of clarity, we have only discussed anti-patterns that arise from role modeling in the case of sortals. However, these anti-patterns are generalizable to also take into account roleMixins. A roleMixin [11] is just like a role, but instead of being played by instances of the same kind (e.g. student-person), it is played by instances of different kinds. Examples of roleMixins include *Customer* and *Insured Item*, the former being played by both people and companies, whilst the latter being played by cars, houses, and paintings.

We highlight that the following studies use these generalized versions of the anti-patterns presented in this chapter. Nonetheless, they are still useful assessments of how frequent and how problematic an anti-pattern is. For more details on the extended version of the anti-patterns, please refer to [21].

We also make emphasize that the version of the *RelRig* anti-pattern presented in this chapter is an evolution of the version used in the empirical studies reported here. The major improvement regards an alteration in the refactoring plan named "Bidirectional dependency". The previous solution suggested that modelers should keep the relation under analysis as a mediation and specify it as a bidirectional dependency. However, this would contradict UFO's definition of objects (substantial individuals), which prevents theses individuals from being existentially dependent of other individuals that are mereologically disjoint from them. To follow this constraint, the refactoring plan was modified to suggest the representation of bidirectional dependencies as part-whole relations.

1.6.1. Frequency Analysis

Our first study measured is anti-pattern frequency, i.e., how often modelers design structures that fit an anti-pattern definition. At this moment, we did not assess if the identified structures were indeed mistakes. To conduct the study, we implemented algorithms to automatically query the models for anti-pattern occurrences, which we later incorporated into the Mentor Editor.

Table 1.6. Frequency evaluation results.

Anti-pattern	Occ.	M.Occ.	$\frac{M.Occ.}{AllMo}$
RelRig	282	37	69%
FreeRole	199	18	33%
MultDep	105	28	52%

To understand how frequent an anti-pattern occurrence is we organize the results in Table 1.6. The meaning of each column is the following: *Occ.* is the sum of all anti-pattern occurrences in all models of the analyzed repository; *M.Occ.* is the number of models with at least one given anti-pattern occurrence in the analyzed repository; $\frac{M.Occ.}{AllMo}$ represents the percentage rate between *M.Occ.* and all models (*AllMo*) in the repository (i.e., 54, in the case of this repository);

As Table 1.6 shows, the three anti-patterns identified here are indeed recurrent modeling decisions. *RelRig*, *MultDep* and *FreeRole* were identified 282, 105 and 119 times, respectively, throughout the 54 models. The fact that these modeling decisions can be found in a significant subset of the inspected models (as shown by $\frac{M.Occ.}{AllMo}$) also corroborates their recurrent nature. Among the three anti-patterns, *RelRig* is the most frequent one, found in 69% of the models, whilst *MultDep* was identified in a little more than half (52%) of the models, and *FreeRole* in roughly a third of them (33%).

1.6.2. Usefulness Analysis

In this second study, we focus on anti-pattern usefulness. The goal is to measure two things: (i) the probability of an occurrence to characterize a domain misrepresentation, i.e., an actual modeling error; and (ii) how often we can predict the solution for an occurrence whenever a problem is identified, i.e., how often our rectification solutions to the problems at hand are adopted by the modelers.

It is not possible to conduct a study of this nature in a fully automatic way, since judging whether an anti-pattern occurrence is an error is a matter of domain knowledge. With that in mind, we chose to conduct the anti-pattern usefulness evaluation as a case study using the MGIC ontology [3]. The first reason we chose this ontology is its size: 3800 classes, 1918 associations, 3616 generalizations, 698 generalization sets, 71 data types, 865 attributes and 149 constraints. The ontology also contains occurrences of all anti-pattern types, having been developed by ten modelers throughout three years. Lastly, the ontology was a product of a governmental project and we had access to the modelers who produced it.

Eight modelers participated in the case study. We assigned sub-domains to each of them, taking into account their knowledge about the domain and its complexity (represented by the number of classes and relations used to formalize it). To guarantee that the modelers would have enough knowledge to analyze the anti-pattern occurrences, we mostly assigned parts of the ontology that they designed. We also encouraged modelers to interact with each other during the case study. Modelers conducted the anti-pattern detection and analysis exclusively using a modeling environment called Menthor Editor.

Together, the participants analyzed 241 occurrences of the anti-patterns presented in this chapter. We summarize the results in Table 1.7. The column *Occ.* stands for the number of analyzed occurrences of a given anti-pattern type, whilst the ones labeled as *Error* and $\frac{Error}{Occ.}$ refer to the total number and percentage of occurrences considered as modeling errors by the participants, respectively. The columns *Refac.*, *Partial* and *Custom* stand for the sum of occurrences the participants fixed using: (i) exclusively our proposed solutions; (ii) a variation of one of our proposed solutions; (iii) and exclusively custom solutions, respectively. $\frac{Refac.}{Error}$ presents the percentage of anti-pattern occurrences fixed exclusively using proposed refactoring plans.

Table 1.7. Usefulness evaluation results.

Anti-pattern	Occ.	Error	$\frac{Error}{Occ.}$	Refac.	$\frac{Refac.}{Error}$	Partial	Custom
RelRig	161	107	66.5%	105	98.1%	1	1
FreeRole	39	23	59.0%	19	82,6%	2	2
MultDep	41	23	56,1%	16	69,6%	6	1
Total	241	153	63,5%	140	91,5%	9	4

By comparing the first two columns of Table 1.7, we have a positive indication that the proposed anti-patterns are indeed error prone structures. In all three cases, the participants considered more than half of the analyzed structures errors. Among the three, *RelRig* seems to be the most pernicious one, since two thirds

of its occurrences were considered problematic.

This case study also provides positive evidence that the refactoring plans are actually viable recurrent solutions. *RelRig*'s refactoring plans were used in almost all scenarios in which a problem was identified, roughly 98% of the cases. *FreeRole*'s predefined solutions were a little less accurate, but still helped participants in 82.6% of the time. The least accurate refactoring plans belong to *MultDep*, even though the results are still positive, namely 69.6%. We hypothesize that this lower predictability is caused by the limited variety of refactoring plans. In fact, we only provide alternatives to move the dependencies down the hierarchy, i.e. to new or existing subtypes, and we neglect the possibility of moving dependencies up the hierarchy.

1.6.3. Combined Results

To provide a conclusion on anti-pattern evaluation, we cross the frequency of occurrence of the anti-patterns (measured the percentage of qualified models containing a given anti-pattern occurrence) with problem-rate (measured by the number of errors divided by number of occurrences) and effectiveness of our rectification plans (quantified as the number of adopted refactoring plans divided by the number of occurrences considered as errors). The higher all these three values are for an anti-pattern, the more useful the anti-pattern is. Anti-patterns that always occur, with a high probability of characterizing a mistake, and with effective associated rectification solutions are more likely to be useful in practice. Less useful anti-patterns, on the other hand, are not those that rarely occur, but instead those that we frequently find but are rarely the source of domain misrepresentations. That is because they require a lot of effort to analyze and provide little gain in model quality. This is not the case of any of the anti-patterns presented in this chapter.

Table 1.8 describes the combined results of the evaluation. Instead of the actual percentages for each measurement, we adopted a discrete scale to classify the values, specified as follows: Very High (80-100%), High (60-80%), Medium (40-60%), Low (20-40%) and Very Low (0-20%).

Table 1.8. Usefulness evaluation results.

Anti-pattern	Frequency	Problem Rate	Predictability
RelRig	High	High	Very High
FreeRole	Low	Medium	Very High
MultDep	Medium	Medium	High

The results show that among the three anti-patterns presented in this chapter, *RelRig* is the one that can contribute more during model validation. It is frequently found, it often points to a modeling error and we are frequently able to precisely automate its associated refactoring solutions. Although *FreeRole* and *MultDep* did not perform as well, they still fit our criteria for useful anti-patterns, since they suggest a still relevant problem rate and their solutions are highly automatable.

1.7. Tool Support

Menthor Editor², formerly known as OntoUML Lightweight Editor (OLED), is an ontology-driven conceptual modeling environment. It provides support for model specification (through ontological patterns), automatic syntax verification, validation (through visual simulation) and code generation for the semantic web (RDF/OWL) and software development (information models). We used the Menthor Editor environment to implement the anti-pattern management functionalities.

In order to make the anti-patterns helpful for model validation, we repeated the strategy adopted in [23]. Firstly, we implemented algorithms to automatically detect occurrences, accessible through the Detection Dialog depicted in Figure 1.4. In the sequence, based on our pre-defined solutions, we implemented wizards to interact with users to support anti-pattern analysis. The wizard for *RelRig* is also depicted in Figure 1.4. Lastly, we implemented algorithms to automatically rectify the model using the input provided during the interaction with the wizard.

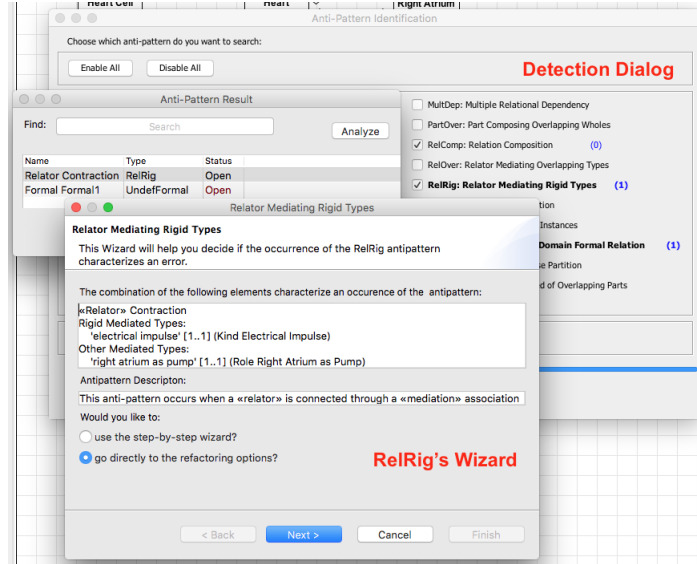


Figure 1.4. Tool support for anti-pattern management implemented in the Menthor Editor.

1.8. Final Considerations

In this chapter, we extended our work on ontological anti-patterns, proposing three new error-prone structures in combination with pre-defined rectification solutions. In particular, the anti-patterns reported here all related to the notion of roles. Role modeling is of fundamental importance in conceptual modeling in general and in ontology engineering in particular. For instance, in the MGIC

²Available at <http://www.menthor.net>

ontology analyzed in one of our empirical studies, «role» is by far the most used construct in the model with 1066 occurrences. Hence, the identification of these anti-patterns and their associated rectification plans as well as their automation in a model-based computational tool constitutes important contributions to the theory and practice of these disciplines.

In order to identify these anti-patterns, we have used two empirical studies. In the first study, by analyzing a model repository of OntoUML models, we investigated the actual frequency of occurrence of these anti-patterns. In the second study, by using a real-world massive governmental ontology, we have investigated how informative are these anti-patterns (how likely they are to spot an actual modeling error) and how effective are the rectification plans associated to them.

Since anti-patterns signal deviations between intended and valid model instances, and since intended model instances only exist in the mind of domain experts, anti-pattern discovery is a human-centric activity. Hence, the anti-patterns currently making our catalog (reported here and in [23]) were discovered in a heavily manual process. As a future work, we intend to address these challenges by studying strategies to (semi) automate the anti-pattern discovery process. For instance, we would like to provide mechanisms that could automatically learn the recurrent correlation between (a) structures in the unintended model instances, (b) structures in the conceptual models that cause them, and (b) solutions provided by the conceptual modelers over (b) in order to rectify the unintended situation identified in (a). Once these strategies are identified and implemented in our computational support, we intend to extend this tool support to be able to automatically identify these anti-patterns across different conceptual models in our model repository. A possibly promising path for investigation in that respect, in the spirit of [1], is the combination of inductive logic learning mechanisms with the counter-example generation capabilities of our model simulation environment (based on Alloy).

Bibliography

- [1] D. Alrajeh, J. Kramer, A. Russo, and S. Uchitel. Automated support for diagnosis and repair. *Communications of the ACM*, 58(2):65–72, 2015.
- [2] J. ao L.R. Moreira, L. Ferreira Pires, M. van Sinderen, and P. Dockhorn Costa. Towards ontology-driven situation-aware disaster management. *Journal of applied ontology*, 10(3-4):339–353, December 2015.
- [3] C. A. M. Bastos, L. Rezende, M. Caldas, A. S. Garcia, S. M. Filho, and J. Castro Junior. Building up a model for management information and knowledge : the case-study for a Brazilian regulatory agency. In *Proceedings of the International Workshop on Software Knowledge, SKY’11*, 2011.
- [4] J. Baumeister and D. Seipel. Anomalies in ontologies with rules. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(1):55–68, 2010.
- [5] W. Brown, R. Malveau, H. McCormick, and T. Mowbray. *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis*. John Wiley & Sons, New York, USA, 1998.

- [6] R. F. Calhau and R. de Almeida Falbo. A configuration management task ontology for semantic integration. In *Proceedings of the 27th Symposium on Applied Computing, SAC '12*, pages 348–353, New York, USA, 2012. ACM.
- [7] H. C. e Silva, R. de Cassia Cordeiro de Castro, M. J. N. Gomes, and A. S. Garcia. Well-founded IT architecture ontology: an approach from a service continuity perspective. In *Proceedings of the 4th Networked Digital Technologies International Conference (NDT'12)*, pages 136–150. Springer, 2012.
- [8] M. Fowler. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, Reading, USA, 1999.
- [9] E. Gamma, R. Johnson, R. Helm, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Boston, USA, 1994.
- [10] N. Guarino and G. Guizzardi. “We need to discuss the relationship”: revisiting relationships as modeling constructs. In *Proceedings of the 27th International Conference on Advanced Information Systems Engineering (CAISE'15)*, pages 279–294. Springer, 2015.
- [11] G. Guizzardi. *Ontological Foundations for Structural Conceptual Modeling*. Centre for Telematics and Information Technology, University of Twente, Enschede, The Netherlands, 2005.
- [12] G. Guizzardi. Ontological patterns, anti-patterns and pattern languages for next-generation conceptual modeling. In *Proceedings of the 33rd International Conference on Conceptual Modeling (ER'14)*, pages 13–27. Springer, 2014.
- [13] G. Guizzardi and T. P. Sales. Detection, simulation and elimination of semantic anti-patterns in ontology-driven conceptual models. In *Proceedings of the 33rd International Conference on Conceptual Modeling (ER'14)*, pages 363–376. Springer, 2014.
- [14] D. Jackson. *Software Abstractions: logic, language, and analysis*. MIT press, 2012.
- [15] A. Koenig. Patterns and antipatterns. *Journal of Object-Oriented Programming*, 8(1):46–48, 1995.
- [16] A. M. Martínez Ferrandis, O. Pastor López, and G. Guizzardi. Applying the principles of an ontology-based approach to a conceptual schema of human genome. In *Proceedings of the 32th International Conference on Conceptual Modeling (ER'13)*, pages 471–478, Berlin, Heidelberg, 2013. Springer.
- [17] J. C. Nardi, R. de Almeida Falbo, J. P. A. Almeida, G. Guizzardi, L. Ferreira Pires, M. J. Van Sinderen, and N. Guarino. Towards a commitment-based reference ontology for services. In *Proceedings of the 17th International Enterprise Distributed Object Computing Conference (EDOC'13)*, pages 175–184. IEEE, 2013.
- [18] OMG. *OMG Object Constraint Language (OCL)*, Version 2.3.1, January 2012.
- [19] M. Poveda-Villalón, M. C. Suárez-Figueroa, and A. Gómez-Pérez. Validating ontologies with OOPS! In *Proceedings of the 18th International Conference Knowledge Engineering and Knowledge Management*, pages 267–281. Springer, 2012.
- [20] C. Roussey, O. Corcho, and L. M. Vilches-Blázquez. A catalogue of OWL

- ontology antipatterns. In *Proceedings of the 5th International Conference on Knowledge Capture*, pages 205–206. ACM, 2009.
- [21] T. P. Sales. *Ontology Validation for Managers*. Universidade Federal do Espírito Santo, Vitória, Brazil, 2014.
 - [22] T. P. Sales, P. P. F. Barcelos, and G. Guizzardi. Identification of semantic anti-patterns in ontology-driven conceptual modeling via visual simulation. In *Proceedings of the 4th International Workshop on Ontology-Driven Information Systems (ODISE'12)*, 2012.
 - [23] T. P. Sales and G. Guizzardi. Ontological anti-patterns: Empirically uncovered error-prone structures in ontology-driven conceptual models. *Data & Knowledge Engineering*, 99:72–104, 2015.