"Is it a Fleet or a Collection of Ships?": Ontological Anti-Patterns in the Modeling of Part-Whole Relations

Tiago Prince Sales[1,2] and Giancarlo Guizzardi[3,4]

[1] Dept. of Information Engineering and Computer Science, University of Trento, Italy
[2] Laboratory for Applied Ontology, ISTC-CNR, Trento, Italy
[3] NEMO Group, Federal University of Espírito Santo, Brazil
[4] Faculty of Computer Science, Free University of Bozen-Bolzano, Italy
tiago.princesales@unitn.it, gguizzardi@unibz.it

Abstract. Over the years, there is a growing interest in employing theories from philosophical ontology, cognitive science and linguistics to devise theoretical, methodological and computational tools for information systems engineering, in general, and for conceptual modeling, in particular. In this paper, we discuss one particular kind of such tools, namely, ontological anti-patterns. Ontological anti-patterns are error-problem modeling structures that can create a deviation between the possible and the intended interpretations of a model. In this paper, we present two empirically elicited ontological anti-patterns related to the modeling of part-whole relations. In particular, these anti-patterns identify possible mistakes in the modeling of collectives (complex entities that have a uniform role-based structure) and functional complexes (complex entities composed of functional parts). Besides identifying these anti-patterns, the paper presents a series of rectification plans that can be used to eliminate their occurrence in models. Finally, we present a model-based computational tool that supports the automated detection, analysis and elimination of these anti-patterns.

Keywords: Ontology-Based Conceptual Modeling, Anti-Patterns, Parthood.

1 Introduction

In recent years, there has been an increasing interest in the application of ontologies in conceptual modeling, including the use of foundational ontological theories to improve the theory and practice of this discipline [1,2]. In these scenarios, ontological theories can play a fundamental role in improving the quality of enterprise-wide conceptual models, improving their quality as artifacts supporting communication, problem-solving, meaning negotiation and, chiefly, semantic interoperability in its various manifestations (e.g., enterprise application integration) [3].

Given the increasing complexity of ontology-driven conceptual modeling, there is an urging need for developing a new generation of complexity management tools for this discipline [1,4]. These include a number of methodological and computational tools that are grounded on sound ontological foundations. In particular, as defended in [1], 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 article focuses on the latter.

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 [1,6]. Such a difference occurs either because the model allows unintended model instances or because it forbids intended ones. We name these configurations Ontological Anti-Patterns.

In this article, we focus on the study of Ontological Anti-Patterns in a particular conceptual modeling language named OntoUML [7]. OntoUML is a language whose meta-model has been designed to comply with the ontological distinctions and axiomatization of a theoretically well-grounded foundational ontology named UFO (Unified Foundational Ontology) [7,8]. 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 several industrial projects in different domains, such as petroleum and gas, digital journalism, complex digital media management, off-shore software engineering, telecommunications, retail product recommendation, and government [8]. A recent study shows that UFO is the second-most used foundational ontology in conceptual modeling and the one with the fastest adoption rate [2]. Moreover, the study also shows that OntoUML is among the most used languages in ontology-driven conceptual modeling (together with UML, (E)ER, OWL and BPMN).

This article can be seen as complementary to our earlier work in [9] and [10]. In [9], we have focused on anti-patterns that are connected to the modeling of material relations (roughly domain associations) and, in [10], on those connected to the modeling of roles. Here, in contrast, we focus on some anti-patterns that emerge when modeling parthood (part-whole) relations. In particular, we focus on anti-patterns that emerge when modelers confuse the ontological unity criteria involved in the modeling of two particular type of parthood relations, namely, the *member of* relation and the *component of* relation.

The contributions of this paper are three-fold. Firstly, we contribute to the identification of two new Ontological Anti-Patterns for conceptual modeling, in general, and for OntoUML, in particular. Secondly, after precisely characterizing these anti-patterns, we propose a set of refactoring plans that can be adopted to eliminate the possible unintended consequences induced by the presence of each of these anti-patterns. Finally, we present an extension for the Menthor Editor[1], an open-source OntoUML model-based editor that: (i) automatically detects anti-patterns in their models; (ii) supports users in exploring whether the presence of an anti-pattern indeed characterizes a modeling error; (iii) automatically executes refactoring plans to rectify the model.

[1] `https://github.com/menthortools/menthor-editor`

The remainder of this article is organized as follows: in Section 2, we briefly elaborate on the modeling language OntoUML and some of its underlying ontological categories, with a particular focus on the modeling of parthood relations; In Section 3, we first briefly present the anti-pattern elicitation method employed here and characterize the model benchmark used in this research; In Section 4, we present the newly elicited Ontological Anti-Patterns with their unintended consequences, as well as possible solutions for their rectification in terms of model refactoring plans; Section 5 elaborates on the extensions implemented in the OntoUML editor taking into account these anti-patterns; Finally, Section 6 presents some final considerations.

2 Ontological and Cognitive Aspects

Parthood is a relation of fundamental importance in conceptual modeling. As such, it is present as a primitive in practically all major conceptual modeling languages and as a micro-theory in all foundational ontologies used in conceptual modeling [7].

Being a cognitively-oriented descriptive ontology, UFO includes micro-theories to address the four kinds of parthood relations generally recognized in Cognitive Science [11,12], namely, the relations of *subquantity-quantity*, *subcollective-collective*, *member-collective* and *component-functional complex*. In UFO, these relations are fully axiomatized and, in their corresponding formal axiomatizations, these relations are characterized w.r.t. formal theories in classical and non-classical mereology [13-17]. In particular, all these relations are shown to conform to the following standard mereological principles: non-reflexivity, asymmetry and the so-called Weak Supplementation Principle (WSP). WSP mandates that if an individual X is part of an individual Y then there must exist at least another individual Z that is mereological disjoint from X and that is also part of Y. In other words, if an individual is mereologically non-atomic (i.e., if it has parts), then it must have at least two disjoint parts.

Being based on UFO, OntoUML has modeling primitives termed *subQuantityOf*, *subCollectiveOf*, *memberOf* and *componentOf* representing these four types of parthood relations, respectively. Moreover, it includes in its metamodel formal constraints representing the axiomatization of these relations according to UFO.

The *subquantity-quantity* is focused on modeling parts of an amount of matter (e.g., alcohol-wine, gin-Martini, ice cream-milkshake) and has been discussed in depth in [17]. This paper focuses on part-whole relations of the three remaining kinds, i.e., the ones involving collectives and their parts (*subCollectiveOf* and *memberOf*) the one involving functional complexes and their parts (*componentOf*).

There is an important difference between an ontological account of parthood such as the one included in UFO and classical mereological theories [13], namely, whilst the latter theories are about a binary relation between the part and the whole, the former theories also address the relations that have to hold between the parts in order for them to form a whole. In other words, we must address the question of what kind of *unity principle* binds the parts together such that they can form a particular whole.

An important ontological distinction between collectives and functional complexes is related exactly to the differentiation between the types of unity principles that form these two types of wholes. In the case of collectives, this unity principle is a uniform relationship (i.e., a relation instance) that holds between all parts and only

those parts [14,15]. Because of the uniformity of this relationship, the collective has a uniform structure, i.e., all its *members* are undifferentiated w.r.t. to the whole. In other words, they can be said to play the same role w.r.t. the whole. Take for example collectives such as a forest, a crowd, a pack of lions or a deck of cards with their corresponding instances of the *memberOf* relation (i.e., tree–forest, person-crowd, lion-pack, card–deck). In all of these cases, the wholes have a uniform structure provided by a uniform unity principle (e.g., a crowd is a collective of person all which are positioned in a particular topologically self-connected spatial location) and their parts are all considered to play the same role w.r.t. the whole (e.g., all persons are equally considered to be *membersOf* the crowd). Having uniform criteria regarding their *membership* does not entail that collectives cannot have differentiated *parts*. However, these parts are of a different kind, namely, they further structure collectives in terms of *subcollectives*. For example, the male portion of the crowd and the female portion of the crowd are *subcollectivesOf* the crowd. Likewise, the teenager portion of the crowd is another *subcollectiveOf* the crowd that *mereologically overlaps* with at least one the former sub-collectives.

In contrast to collectives, functional complexes are unified by a *functional architecture* formed by a chain of *functional dependence* relations [16]. In a functional complex, there is a differentiation of the roles played by the different parts. Moreover, by playing these different functional roles, the parts contribute in complementary manners to the functionality of the whole. Take for example functional complexes such as a circulatory system, a car, a computer network, or an organization and their corresponding *componentOf* relations (i.e., heart-circulatory system, engine-car, router-computer network, presidency-organization). In all these examples, the parts play particular functional roles contributing in specific ways to the functionality of the whole (e.g., the heart plays the functional role of pumping blood w.r.t. to the circulatory system such that the circulatory system cannot function as such without having a component play that particular role of blood pump) [16].

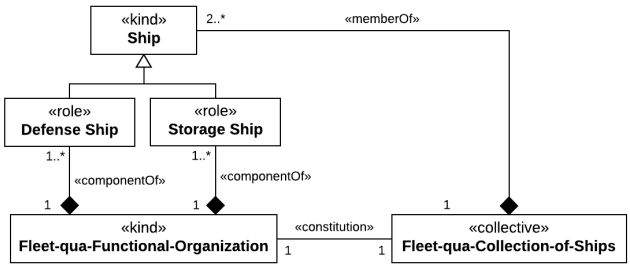


Fig. 1. Fleet as a functional complex *versus* fleet as a collection.

Finally, it is important to highlight that many natural language terms exhibit a case of *systematic polysemy* [18] in referring both to collectives and functional complexes. For example, take the case of a fleet, as discussed by [11,12]. In the case that all ships of a fleet are conceptualized as playing solely the role of a *memberOf* a fleet, then the term fleet can be said to refer to a collection. In contrast, if a fleet is conceptualized from a functional perspective in which roles are further specialized in *leading ship*, *defense ship*, *storage ship* and so forth, the fleet term refers to a functional complex.

In other words, the term fleet seems to refer in a polysemic manner to two different entities: one that is an organizational entity/functional complex that has a functional architecture in which parts play a number of differentiated roles; another that is just a collection of ships. On one hand, these two entities are distinct, following different identity and unity principles (e.g., while replacing an individual ship creates a different *collective of ships* it does not alter the identity of the *fleet-qua-functional-organization*). On the other hand, they bear a particular relation to each other, namely, a relation of *constitution* [7], i.e., the *fleet-qua-functional-organization* is constituted by the *fleet-qua-collection-of-ships,* as depicted in Fig. 1.

Following the ontological distinctions put forth by UFO, OntoUML countenances three different stereotypes that can be applied to types of substantial entities in the domain, depending on the nature of their unity criteria: «kind» for types of functional complexes, «collective» for types of collectives, and «quantity» for types of quantities. The types marked with these three stereotypes represent what the modeler deems to be the *kinds* of entities in the domain (in the ontological sense). As such, these types aggregate essential properties for their instances. For this reason, they are static (i.e., modally rigid types), meaning that they classify their instances in all possible situations. Rigid types that specialize those former three types are stereotyped as «subkind» (e.g, the «subkind» Man specializes the «kind» Person); dynamic types specializing them are either stereotyped as «role», when their dynamic classification condition is a relational one (e.g., student, husband, father), or «phase», in case their dynamic classification condition is an intrinsic one (e.g., teenager, puppy or living person); abstract types (aka *dispersive types* [7]) that classify instances of more than one *kind* (i.e., more than one type stereotyped as «kind», «collective» or «quantity», or any combination of these) are stereotyped as «category» (in case they are rigid abstract types, e.g., the type *Physical Object* rigidly classifying entities of kinds people, buildings, dogs, car, etc.), «roleMixin» (in case they are dynamic and relational abstract types, e.g., the type *Customer* classifying entities of kinds people and organization) or «mixin» (an abstract type that is static to some instances and dynamic to others, e.g., the type Insured Item classifying rigidly things of the type *Car* and dynamically things of the kinds *Trip* and *Building*).

For an in depth discussion and formal characterization of UFO and OntoUML, one should refer to [7]. In particular, for ontological and cognitive aspects and formal characterization of collectives and functional complexes in UFO, as well as the corresponding OntoUML profiles for the *memberOf/subCollectiveOf* and *componentOf* relations, one should refer to [15] and [16], respectively.

3 Methods and Materials

Our approach to identify ontological anti-patterns is an empirical qualitative analysis. It starts with the selection of a model for analysis, which is followed by the identification of relevant model 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 in order to uncover possible problems. We conduct this activity using visual model simulation [1,6]. This simulation consists in converting OntoUML models into

Alloy [19] 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 [1,6] according 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 that is the case, we propose a possible solution to rectify the model and register it as a problem-solution pair. With a modified model, we go back to step three. This iteration is repeated until no more problems can be identified in that 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/refactoring plans.

Our empirical analysis for uncovering anti-patterns was performed using a repository of 54 models[2]. Out of these, 11 models were developed in the context of academic research without industry collaboration. An example is The Configuration Management Task Ontology [20], a product of a Masters dissertation. Furthermore, 7 models had total or partial participation of private companies and/or governmental organizations, the most significant being the MGIC Ontology [21], developed within a re-search project with a regulatory agency responsible for controlling ground transportation services in Brazil.

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 [22] 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 [23]. The repository also contains 8 models (13%) designed for knowledge-based applications, 6 (10%) whose main intention was to support semantic interoperability between systems and/or organizations, and only 2 (3%) for the purpose of enterprise modeling. For the remainder 26 models (42%), there is no information w.r.t. this aspect of classification.

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. Most models (35 out of 54) were developed individually, whilst 15 were the product of a collaboration between 2-4 people, and 4 involved 7-10 people.

The two anti-patterns discussed in next section appeared in 37,04 % of the models with 142 occurrences (*HomoFunc*, see Section 4.2) and in 12,96% of the models with 60 occurrences (*HetColl*, see Section 4.1).

---

[2] The models we used in our research, with an exception of a few (due to non-disclosure agreements), are available at `http://www.menthor.net/model-repository.html`.

4 Ontological Anti-Patterns

4.1 Heterogeneous Collective (HetColl)

As we discussed in Section 2, a collective is an entity whose parts (members) play the same role w.r.t. whole. If we say that a troupe is a collection of artists, we are implying that all artists play merely the generic role of being part of the troupe. Conversely, functional complexes are entities whose parts play different roles w.r.t. whole, thus making different contributions to the behavior of the whole. For instance, the CPU is a functional part of a computer, as well as the hard-drive, since the former is responsible for processing operations, whilst the latter is responsible for storing non-volatile data. As discussed in [7], sometimes, different conceptualizations can articulate a notion in reality as a functional complex or as a collective. As previously mentioned, one conceptualization can articulate a fleet as a functional complex, in which different ships play different functional roles, while another conceptualization can articulate it as merely a collection of ships.

In OntoUML, collectives can be further refined into sub-collections. Again, although defining collective parts of super-collective provides further structure to the whole, it still does not differentiate roles played by their members. For instance, the troupe could be refined into sub-collections of singers, dancers and actors, whose members are the artist who can sing, dance and act, respectively. In this case, the principle unifying these sub-collections is just a strengthen of the common principle that unifies the collection in the first place [15] and, thus, all the members of the collection are still undifferentiated w.r.t. the whole. In other words, the whole regards them merely as *members*.

The Heterogeneous Collective (HetColl) anti-pattern identifies collectives that are connected via membership relation to *members* classified under different types. This can be an indication that the modeler either confused the collection and functional complex interpretations of the same notion or confused the membership and sub-collection relations. By analyzing the models in the OntoUML repository and by discussing their intended semantics with their respective modelers, we have notice that whenever a collective noun (like fleet, group, pack) is used, modelers are most likely to represent it as a collective, without fully analyzing the subtleties of the particular conceptualization at hand.

The key aspect to successfully analyze this anti-pattern is to identify the nature of the unity criteria connecting the parts that form the whole. If one concludes that the parts in fact play different roles w.r.t the whole, the refactoring plan is to change the ontological category of the whole to a functional complex (if necessary, also change the ontological category of parts) and then to change the stereotype of the meronymic relations to *componentOf*, instead of *memberOf*. Alternatively, if one concludes that the members indeed play the same role regarding the whole, we propose to make this position explicit by creating a type as the direct parent of all current part types and then merge all *memberOf* relations into a new that is connected to this newly created supertype. Yet a third alternative is presented when the modeler concludes that the types representing the members are in fact sub-collections, i.e., they are refinements

of internal structure of the collective whole [15]. To rectify this case, one must change the stereotypes of the *memberOf* relations to *subCollectionOf* and, if necessary, adjust the ontological category of the parts to be that of a collection.

Table 1. Characterization of the *HetColl* anti-pattern.

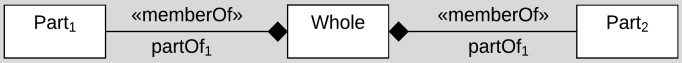
Acronym	**Name**
HetColl	Heterogeneous Collective
Description	
A collective connected to multiple types of member through «memberOf» relations suggests a misrepresentation of the concepts w.r.t. to the ontological nature of the whole and the meronymic relations.	
Pattern Roles	

#	**Name**	**Allowed Sterotypes**
1	Whole	«collective», «subkind», «phase», «role», «category», «roleMixin», «mixin»
2..*	$partOf_n$	«memberOf»
2..*	$Part_n$	«kind», «collective», «subkind», «phase», «role», «category», roleMixin», «mixin»

Additional Constraints

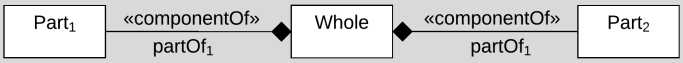
The *Whole* should be: (i) stereotyped as collective; (ii) stereotyped as subkind, role or phase and be a direct or indirect subtype of another type stereotyped as collective; or (iii) be stereotyped as mixin, category or roleMixin and have all its direct or indirect subtypes meeting conditions (i) or (ii)

Generic Example

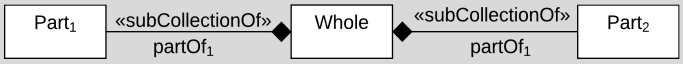


Refactoring Plans

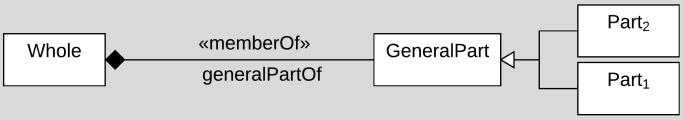
1. Change parts to functional parts: Change the nature of *Whole* to functional complex and change the stereotype of every $partOf_n$ to «componentOf».



2. Change parts to sub-collections: Change the stereotype of every $partOf_n$ to «subCollectionOf». If a $Part_n$ is not a collection, this should be rectified in the model.



3. Set a generic membership: Create *GeneralPart*, a common direct supertype of every $Part_n$; remove every $partOf_n$ from the model; and create *generalPartOf, a* new «memberOf» from *Whole* to *GeneralPart*. The stereotype ascribed to *GeneralPart* can be derived from the stereotype of $Part_n$.



Note that in both the first and third refactoring plans described, whenever there is a need to ontological category of either a part type or a whole type, the following strategy can be adopted. If (1) a type A is erroneously stereotyped as a «kind» or a «quantity», then A should be represented as a «collective». However, if (2) A is stereotyped as «subkind», «role» or «phase» and (directly or indirectly) specializes a «kind» or a «quantity» type S, one can: (2.1) change the stereotype of S to «collective»; (2.2) select another «collective» in the model and make A specialize it; or (2.3) create a new «collective» be the supertype of A. Finally, if (3) P is stereotyped as a «category», «mixin» or «roleMixin», strategies (1) and (2) should be adopted for every subtype of P.

An example of the occurrence of *HetColl* is depicted in Fig. 2 below. This model is an adaptation of a fragment extracted from a governmental conceptual model in the domain of agricultural protection. The fragment describes a particular type of work group, named Technical Administrative Support Group, which has employees that play the roles of technical and/or administrative support. The cardinalities constraints defined in the association ends of parts show that this type of work group requires employees performing different duties. The requirement of the presence of people playing these different roles indicates that the work group should be really modeled a functional complex instead of a collective. As it is typical in these cases, there is another implicit entity, namely, the staff of the work group, which at each point in time constitutes the Work Group as a functional complex. However, the Work Group itself and its staff are ontologically distinct entities as these are associated with different identity criteria (e.g., while changing a member of the staff creates a different staff, it can still be the same Work Group, just then constituted by a different staff) [7].

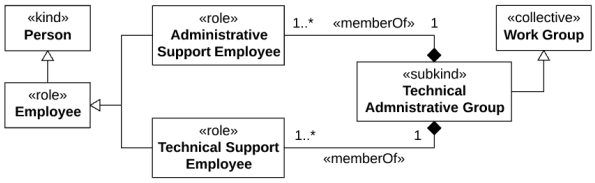


Fig. 2. HetColl occurrence in a fragment of a governmental model.

4.2 Homogeneous Functional Complex (HomoFunc)

The Homogeneous Functional Complex (*HomoFunc*) anti-pattern is the counterpart of the *HetColl* anti-pattern. As discussed in Section 2, functional complexes have heterogeneous structures, such that its parts play different functional roles w.r.t. the whole. Therefore, when a modeler describes a functional complex, it is usually expected that she would represent multiple *componentOf* relations connected to such type to account for the diversity of such functional roles.

Table 2. Characterization of the *HomoFunc* anti-pattern.

Acronym	**Name**
HomoFunc	Homogeneous Functional Complex

Description

A functional complex connected to a single part through a «componentOf» relation suggests that all instances of the part play the same role w.r.t. their whole, a homogeneous structure that does not characterize a functional complex.

Pattern Roles

#	Name	Allowed Sterotypes
1	whole	«kind», «subkind», «phase», «role», «category», «roleMixin», «mixin»
1	part	«kind», «subkind», «phase», «role», «category», «roleMixin», «mixin»
1	partOf	«componentOf»

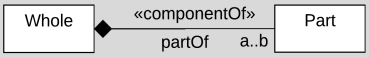
Additional Constraints

1. Both *Whole* and *Part* should be: (i) stereotyped as kind; (ii) stereotyped as subkind, role or phase and be a direct or indirect subtype of another type stereotyped as kind; or (iii) be stereotyped as mixin, category or roleMixin and have all its direct or indirect children meeting conditions (i) or (ii)

2. The lower bound multiplicity of *partOf*'s association end connected to *Part* is greater than one

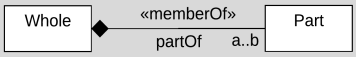
3. *Whole* is not a subtype of another class who has other types of part

Generic Example

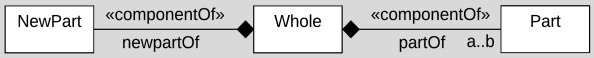


Refactoring Plans

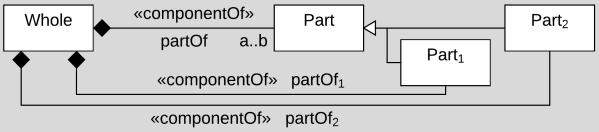
1. Change to membership: change the nature of *Whole* from a functional complex to a collection and change the stereotype of *partOf* from «componentOf» to «memberOf»



2. Add functional parts: specify additional functional parts for *Whole*



3. Add subtypes for *Part*: specify subtypes for *Part* and connect them to *Whole* through additional «componentOf» relations. If the original *partOf* relation is kept in the model, the added relations must subset, redefine or specialize it (see discussion in [24]).



In OntoUML, a type represents a functional complex when: (i) it is stereotyped as a «kind»; (ii) it is a subtype of kind (i.e., as a «subkind», «role» or «phase»); (iii) it

represents a non-sortal type (i.e., «category», «mixin» or «roleMixin») and all its (direct or indirect) sortal subtypes satisfy conditions (i) or (ii). An occurrence of the *HomoFunc* anti-pattern is observed in a model when there is a model fragment representing a homogenous structure of a functional complex, i.e., there is a «kind» type connected through a single *componentOf* relation to one single type of part.

In our empirical investigation, we observed that a common reason for an occurrence of *HomoFunc* is when a modeler mistakenly represents a functional complex while actually intending to represent a collective. This can happen because alternative conceptualizations can ascribe different interpretations for the same term in the domain (see discussion on section 3.1). For this case, we propose the following refactoring plan (see Table 2): one should transform the functional parthood relation at hand (a *componentOf*) in a membership relation (a *memberOf*). Then, one should change the ontological category of the whole type to that of a collection.

In a second situation, we have that a modeler actually intended to represent a heterogeneous structure for the whole. In this case, the modeler should refine the model to include additional types of part. This can be accomplished in two different albeit non-exclusive ways. First, through the specification of new types of functional parts, i.e., types that bear no taxonomic relations to the type already present in the model representing the part (see refactoring plan 2 in Table 2). Second, through the creation of subtypes of the single functional part, alongside with the additional corresponding *componentOf* relations (see refactoring plan 3 in Table 2).

In table 2, the constraint number 2 for the characterization of this anti-pattern expresses that the parthood relation represented should satisfy the weak supplementation axiom (one of the most fundamental axioms of parthood, see [13-17]). Otherwise, the situation would indicate simply an incomplete model and would not exemplify an occurrence of this anti-pattern. Constraint number 3, instead, exclude the situation in which role differentiation is guarantee by additional parthood relations inherited from a possible supertype of the type representing the whole.

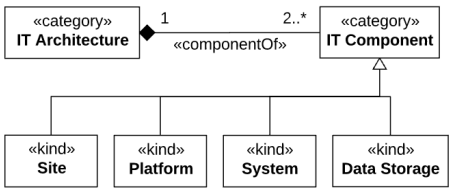


Fig. 3. HomoFunc **o**ccurrence in a fragment of an IT infrastructure model.

An example of the occurrence of *HomoFunc* is depicted in Fig. 3. The model fragment is extracted from the PAS 77 ontology [25], a model in the domain of IT architecture. Notice that the IT Architecture type is defined as being solely composed of IT Components, which in turn can be sites, platforms, operating systems and data storage units. In its original form, the model suggests that all architectural parts are equal w.r.t. the IT Architecture and, hence, that an IT architecture is simply a collection of IT components. If this is the intended semantics, a more suitable formalization would be to represent IT Architecture as a *collective* connected to its parts by a *mem-*

berOf relation (as described by the first refactoring plan on Table 2). Conversely, if this would not the intended semantics, the model would be more accurate if the specific parthood relations between IT Architecture and its different functional components were made explicit (following a the third refactoring plan proposed in Table 2).

5 Tool Support

The Menthor Editor, formerly known as OntoUML Lightweight Editor (OLED), is an open-source ontology-driven conceptual modeling environment. A full support for anti-pattern management has been implemented in this editor. Following the strategy adopted in [9], these anti-pattern management functionalities include anti-pattern detection, analysis (via a wizard-based feature) and elimination (using the rectification plans proposed here). In other words, by employing the explicitly defined MOF metamodel on which this editor is based, we have: firstly, implemented algorithms to automatically detect anti-pattern occurrences, accessible through a detection dialog window (see example in the left part of Fig.3); in sequence, based on our pre-defined solutions (rectification plans), we implemented wizards to interact with users to support anti-pattern analysis (the right part of fig.3 depicts a wizard for the HetColl anti-pattern); finally, we implemented algorithms to automatically rectify the model using the input provided during the interaction with this wizard. In the Fig. 4, we have used these automated functionalities to evaluate the model of Fig. 2.

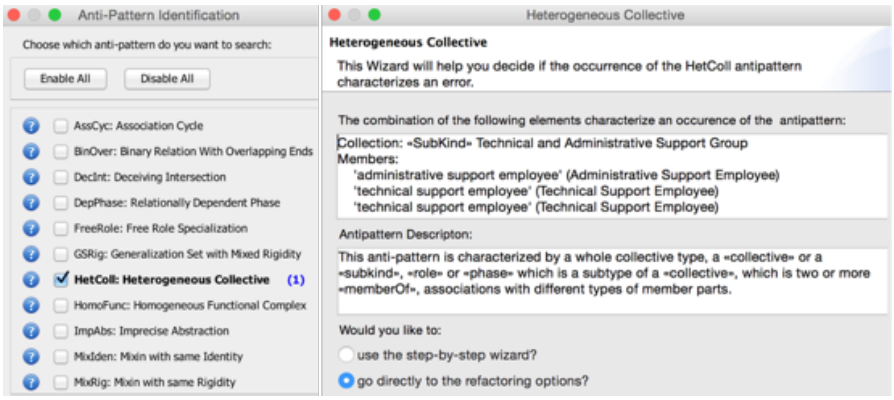


Fig. 4. Tool support for anti-pattern management.

6 Final Considerations

In this paper, we extended our work on ontological anti-patterns, proposing three new error-prone structures in combination with pre-defined rectification solutions. In particular, we focused on anti-patterns related to the modeling of parthood (collectives and functional complexes) in conceptual modeling. Parthood is of fundamental importance in conceptual modeling, in general, and in areas such as Enterprise Model-

ing, Economy and Finance, Manufacturing, Life Sciences, among others, in particular. For this reason, 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.

We emphasize that it is not among our goals in this paper to defend particular modeling choices for specific concepts such as fleet, IT architecture or Work Group. In other words, we have no stand here in whether, in general, concepts such as these are better represented as *functional complexes* or *collectives*. The adequacy of a representation choice over another (or even both choices used simultaneously, following a modeling pattern such as the one of Fig.1) depends exclusively on the rationale of a particular model and its underlying conceptualization. Nonetheless, the anti-patterns proposed in this paper are able to identify situations in which modelers recurrently make mistaken representation choices in that respect, i.e., choosing to use a collection to represent what is a functional complex in the domain, or vice-versa.

References

1. Guizzardi, G., Ontological patterns, anti-patterns and pattern languages for next-generation conceptual modeling. Proceedings of the 33rd International Conference on Conceptual Modeling (ER'14), pages 13–27. Springer, 2014.
2. Verdonck, M., Gailly, F.: Insights on the Use and Application of Ontology and Conceptual-Modeling Languages in Ontology-Driven Conceptual Modeling. In: Proceedings of ER 2016. LNCS, Vol. 9974, 83-97. Springer (2016).
3. Nardi, J.C., Falbo, R.A, Almeida, J.P.A., Foundational Ontologies for Semantic Integration in EAI: A Systematic Literature Review. I3E 2013: 238-249 2014.
4. Guizzardi, G., Theoretical Foundations and Engineering Tools for Building Ontologies as Reference Conceptual Models, Semantic Web Journal, Editors-in-Chief: Pascal Hitzler and Krzysztof Janowicz, IOS Press, Amsterdam, 2010.
5. Koenig, A., Patterns and antipatterns. Journal of Object-Oriented Programming, 8(1):46–48, 1995.
6. Benevides, A.B. et al., Validating modal aspects of OntoUML conceptual models using automatically generated visual world structures, Journal of Universal Computer Science, Special Issue on Evolving Theories of Conceptual Modeling, Editors: Klaus-Dieter Schewe and Markus Kirchberg, 2010.
7. Guizzardi, G. Ontological Foundations for Structural Conceptual Modeling. Telematics Institute Fundamental Research Series, Enschede, The Netherlands, 2005.
8. Guizzardi, G. et al., Towards Ontological Foundation for Conceptual Modeling: The Unified Foundational Ontology (UFO) Story, Applied Ontology, v.10, IOS Press, 2015.
9. Sales, T.P., Guizzardi, G., Ontological anti-patterns: Empirically uncovered error-prone structures in ontology-driven conceptual models. DKE, 99:72–104, 2015.
10. Sales, T.P., Guizzardi, G., Anti-patterns in Ontology-driven Conceptual Modeling: The Case of Role Modeling in OntoUML, Ontology Engineering with Ontology Design Patterns: Foundations and Applications, A. Gangemi, P. Hizler, K. Janowicz, A. Krisnadhi, V. Presutti (Editors), IOS Press, The Netherlands, 2016.
11. Pribbenow, S. Meronymic Relationships: From Classical Mereology to Complex Part-Whole Relations, The Semantics of Relationships, Kluwer Academic Publishers, 2002.
12. Gerstl, P. and Pribbenow, S., Midwinters, End Games, and Bodyparts. A Classification

of Part-Whole Relations, Intl. Journal of Human-Computer Studies 43: 865-889, 1995.
13. Varzi. A.C.: 'Parts, wholes, and part-whole relations: The prospects of mereotopology'. Journal of Data and Knowledge Engineering, 20:259–286, 1996.
14. Simons, P.M., Parts: An Essay in Ontology, Clarendon Press, Oxford, 1987.
15. Guizzardi, G., Ontological Foundations for Conceptual Part-Whole Relations: The Case of Collectives and their Parts, 23rd International Conf. on Advanced Information System Engineering (CAiSE'11), London, UK.
16. Guizzardi, G. The Problem of Transitivity of Part-Whole Relations in Conceptual Modeling Revisited, 21st International Conference on Advanced Information Systems Engineering (CAISE'09), Amsterdam, The Netherlands, 2009.
17. Guizzardi, G. On the Representation of Quantities and their Parts in Conceptual Modeling, Proceedings of FOIS 2010, IOS Press, Toronto.
18. Ravin, Y., Leacock, C., Polysemy: Theoretical and Computational Approaches. Oxford University Press, USA, 2002, p. 240.
19. Jackson, D. Software Abstractions: logic, language, and analysis. MIT press, 2012.
20. Calhau, R.F., Falbo, R.A., 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.
21. Bastos, C.A.M. et al., Building up a Model for Management Information and Knowledge: The Case-Study for a Brazilian Regulatory Agency, in International Workshop on Software Knowledge (SKY), 2011.
22. Nardi, J.C. et al. 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.
23. Ferrandis, A.M.M. et al., Applying the principles of an ontology-based approach to a conceptual schema of human genome. Proceedings of ER 2013, Hong Kong.
24. Costal, D. et al., Formal Semantics and Ontological Analysis for Understanding Subsetting, Specialization and Redefinition of Associations in UML, 30th International Conference on Conceptual Modeling (ER 2011), Brussels, Belgium, 2011.
25. Silva, H.C., de Castro, R.C.C., Gomes, M.J.N., Garcia, A.S., Well-founded IT architecture ontology: an approach from a service continuity perspective. 4th Networked Digital Technologies International Conference (NDT'12), p.136–150. Springer, 2012.