# On the Identification and Representation of Ontology Correspondence Antipatterns

Anselmo GUEDES [1], Fernanda BAIÃO and Kate REVOREDO
*Department of Applied Informatics, Federal University of the State of Rio de Janeiro – UNIRIO, Brazil*

**Abstract.** Semantic data integration and application interoperability are recognized challenges in distributed scenarios, and they have long being addressed by ontology alignment techniques, which try to find correspondences between entities that are semantically equivalent in distinct ontologies. However, ontology alignment is still a costly and difficult task, considering the existence of large-scale scenarios and complex domains. This work introduces the concept of a correspondence antipattern, which is essentially a set of generic correspondences that represent an incorrect alignment and can be used as a predefined template to help identify incorrect correspondences between two existing ontologies and thus refine ontology alignments. We also introduce a methodology that assists the identification and construction of ontology correspondence antipatterns, and evaluate it through a case study using datasets from the last three campaigns of OAEI, the Ontology Alignment Evaluation Initiative.

**Keywords.** ontology matching, correspondence antipatterns, alignment problem.

## Introduction

As the research and practice on Ontology become more popular and evolve, several ontology artifacts arise for the same universe of discourse. However, they differ among each other in several perspectives, such as distinct representation languages (syntactic heterogeneity), variations in names referring to the same entity (terminological heterogeneity), different conceptualizations for the same domain (conceptual heterogeneity) and entities being perceived differently (semiotic heterogeneity)[19]. The Ontology Matching area [19] [23] deals with all these problems, being considered by many authors the key element for heterogeneity reduction between ontologies.

The Ontology Matching task consists in identifying the correct correspondences among entities of multiple ontologies, which it is a necessary condition for establishing the interoperability among them [23]. Techniques used to identify the correspondence between the entities of two ontologies include the analysis of subsumption between classes and the similarity between the entity names [19]. However, current results from state-of-the-art techniques are neither complete nor precise, i.e., they are not able to identify all existing correspondences between two ontologies and sometimes suggest

[1] Corresponding Author: anselmo.guedes@uniriotec.br

correspondences that may be incorrect [26]. With regard to precision errors, suggesting an incorrect correspondence may lead to either logical or ontological incompatibilities.

On the other hand, in the context of software development, antipatterns are considered a valuable tool for the identification of bad or incorrect practices. Antipatterns prevent or hamper the good conduct of the software development and maintenance processes. According to Sales et al. [12], an anti-pattern is "just like pattern, except that instead of a solution it gives something that looks superficially like a solution, but is not one". It is, thus, a recurrent decision for a specific scenario that usually results in more negative consequences than positive ones [11]. Once the instantiation of an anti-pattern is identified in a solution, this solution should be refactored so as to become an appropriate solution.

In the context of ontology matching, we claim that those bad solutions consist of incorrect (including missing) correspondences. A correspondence antipattern is then a matching model for identifying incorrect correspondences that may occur repeatedly in ontology matching processes. The results from an ontology matching approach could be improved by looking for instances of the correspondence antipatterns within the results and proposing a refactoring of the identified problem, so that the problematic situation is removed.

In this work, we introduce the concept of correspondence antipatterns and show a methodology for identifying and computationally representing them. Furthermore, we apply this methodology in some scenarios of the OAEI (Ontology Alignment Evaluation Initiative) and identified one correspondence antipattern.

This work is divided as follows: Section 1 presents related work in the literature; Section 2 defines the ontology matching problem. Section 3 presents the basic concepts of design patterns and antipatterns. Section 4 introduces the concept of a correspondence antipattern and a methodology for its identification and representation. Section 5 presents results of a case study on the OAEI, while and, finally, Section 6 presents the final considerations of this work.

## 1. Related Work

Many of the existing research have given focus on identification and specification of antipatterns in general. In the literature about antipatterns, Brown et al. [3] is particularly relevant. This work identified antipatterns that can be detected not only in the architecture and design of systems, but also in project management software. Other research [4] [5] identified performance antipatterns as their main concern, because impact directly on the quality of the software.

Some research has been carried out to provide formalism to antipattern specifications. For example, Ballis et al. [7] propose a new visual language for describing patterns and antipatterns. This language is an extension of UML with some new graphics, so that patterns and antipatterns can be specified in a more rigorous. More recently, Stamelos [8] proposed the use of Bayesian Belief Networks, Ontologies, arrays of structures and Design social networks as tools to formally represent antipatterns Software project management.

In ontology research, Ontology Design Patterns (ODPs) are an emerging approach that favors the reuse of encoded experiences and good practices. ODPs are modeling solutions to solve recurrent ontology development problems [9]. According to Falbo et al. [10], patterns in Ontology Engineering are still in infancy when compared with

Software Engineering, where patterns have been used for a long period. The earliest works addressing patterns in Ontology Engineering are from the beginning of the 2000s. Sales and colleagues [11] present semantic antipatterns for ontology engineering. These antipatterns capture error-prone modeling decisions, which can result in the creation of models that allow for unintended model instances (representing undesired state of affairs). The authors present antipatterns that have been empirically elicited by validating the ontology conceptual models via visual simulation. The proposed antipatterns do not comprise corresponding relations between entities, which are crucial for the Ontology Matching task.

In the context of Ontology Matching, correspondence patterns were proposed by [12] and are essentially correspondences and sets of correspondences with generic entities. They help find more precise correspondences than simply relating one entity to another one. Each correspondence pattern is a generic solution to a problem of alignment. Scharffe [12] proposed a library of correspondence patterns that represent solutions to different recurring mismatches which are quite hard for matchers using usual matching techniques. However, correspondence patterns do not help in scenarios where the ontology matching technique returns an ontology alignment including an incorrect correspondence. ASMOV (Automated Semantic Matching of Ontologies with Verification) is an algorithm that uses lexical and structural characteristics of two ontologies to iteratively calculate a similarity measure between them, derives an alignment, and then verifies it to ensure that it does not contain semantic inconsistencies [27]. ASMOV is designed to combine a comprehensive set of element-level and structure-level measures of similarity with a technique that uses formal semantics to verify whether computed correspondences comply with desired characteristics.

Padilha [13] proposes design patterns and antipatterns for ontology alignment using top-level ontologies. The proposed design patterns were built based on OntoUML [14], an ontology modeling language which considers the ontological distinctions and axiomatic theories proposed in Unified Foundational Ontology (UFO). UFO was proposed by Guizzardi [28] and meets axiomatic theories that deal with the main categories of concepts used in conceptual modeling [29]. In [13], some patterns are formally defined focusing on the analysis of the OntoUML modeling constructs; however, there was no intention to specify a generic definition for antipatterns in ontology matching, or to address the problem of their identification and operationalization.

## 2. The Ontology Matching Problem

Ontology is an explicit specification of a conceptualization, in which classes, attributes, relationships and instances are considered as first class elements [12]. Formally, an ontology may be represented as a tuple $O = <C, R, P, I, A>$, where $C$ is a set of classes (concepts of the domain), $R$ is a set of relationships between classes in $C$ (also called object properties in some languages), $P$ is a set of data properties (a specific type of relation whose domain is a class and the range is a data type), $I$ is a set of class instances (concrete objects) and $A$ is a set of axioms. Meilicke [1] also defines a signature $S$ of $O$ as $S = <C, R, P, I>$.

Ontology matching identifies correspondences between the entities of multiple ontologies, and it is a necessary condition to establish interoperability between them

[23]. According to Euzenat [19], technically the ontology matching process occurs by taking two ontologies $O$ and $O'$ as input, optionally adding a set of resources $r$, a set of parameters $p$ and a preliminary alignment $A$. The result of this process is an alignment $A'$ between the ontologies $O$ and $O'$, represented as $A' = f(O, O', A, p, r)$.

Basically, ontology matching is a process in which semantic links between entities of ontologies are established. Each semantic link is called a correspondence. A set of correspondences is called an alignment. According to Sváb-Zamazal [18], consider two ontologies $O$ and $O'$, and a function $Q$ that defines corresponding entity sets $Q(O)$ and $Q(O')$ in which $Q(O) \subseteq O$ and $Q(O') \subseteq O'$. A correspondence $D$ between $O$ and $O'$ is a quadruple $\langle e, e', r, n \rangle$ so that $e \in Q(O)$, $e' \in Q(O')$, $r$ is the semantics of the correspondence (for example, equivalence), and $n \in [0,1]$ is a confidence value. An alignment $A$ between $O$ and $O'$ is a set of correspondences between $O$ and $O'$.

When two classes (or sets of classes) do not match, we can affirm, in turn, that their sets of possible instances is not equivalent, so we have the following definition for a non-correspondence [13]: Consider the classes $e$ and $e'$ and their sets of possible instances in all possible worlds I$e$ and I$e'$ respectively (and analogously the sets of classes C and C' and their sets $I_C$ and $I_{C'}$). If $e'$ and $e$ do not match by the relation $r$ (either equivalence or specialization), then I$e' \not\subset$ I$e$. In other words: $\neg (e, e', r, n) \leftrightarrow$ I$e' \not\subset$ I$e$ and $\neg$ (C, C', $r$, $n$) $\leftrightarrow I_{C'} \not\subset I_C$. A non-equivalence correspondence between classes may be defined as: $\neg\langle e, e', equality, n\rangle \leftrightarrow$ I$e' \neq$ I$e$.

## 3. Design Patterns and Antipatterns

When specialists work on a particular problem, it is not very common that new proposals are completely different from existing solutions. Often, solutions to similar problems are retrieved and the essence of the solution is reused for the resolution of the new problem [15]. Patterns assist in building a collective experience based on the skills of software engineers. They capture existing and proven experience in software development and help to promote good design practices. Each pattern deals with a specific problem and recurring design or implementation of software. A software design pattern describes a particular design problem (project) that arises in specific contexts and presents a proven generic schema for the solution [15].

On the other hand, an antipattern is a description of a given solution to a common problem that generates, definitely, negative consequences. When properly documented, an antipattern describes the overall solution, the primary causes that lead to problematic solution, symptoms that describe how to recognize the antipattern, consequences of using this solution and a refactored solution that can change the antipattern in a most appropriate solution [3]. Antipatterns are new forms of patterns. A fundamental difference between a pattern and an antipattern is that the solution adopted which the antipattern identifies, presents negative consequences on their use. Some consequences may appear immediately (symptoms) and some might appear only in the future (consequences). To be truly useful, antipatterns should present an appropriate solution to the problem identified [3], in the other words, the refactored solution. According to [16], refactoring is the process of changing a software system in such a way that they do not alter the external behavior of the code yet improves its internal structure. Is a disciplined way to clean code that minimizes the chances of introducing errors.

The use of templates is what makes the design patterns and antipatterns different from other forms of technical discussions. The models ensure that important issues are answered for each pattern and antipattern. According to Brown [3], antipatterns can be specified in three different ways: (i) Pseudo Antipattern Template, in which the author only textually describes a bad solution; (ii) Mini Antipattern Template is the most basic form of the antipattern, consisting only of its name, the problem identified and refactored solution; and (iii) Complete Antipattern Template, which consists of a detailed description and specific features of the antipattern. We adopt the complete antipattern model as detailed in Table 1.

**Table 1.**Complete Antipattern Model

| Antipattern Item | Short Description |
|---|---|
| **Name** | The name of the antipattern. |
| **Refactored solution type** | A solution may be given at the level of software, technology, process and roles. Software-level solution indicates that new software is created for the solution. Technology indicates that the solution implies the acquisition of new technology or a product. Process indicates that must follow a solution from the process involved. Role indicates that the solution implies the attribution of responsibilities to an individual or group. |
| **Root cause(s)** | The general cause(s) of the antipattern. |
| **Unbalanced forces** | The primary forces that are ignored, misused or used too much in this antipattern. |
| **Background** | Examples of where the problem may occur or general information that may be useful for better understanding the antipattern. Is optional. |
| **Antipattern general form** | A diagram or schema that identifies general characteristics of the antipattern. |
| **Symptoms and/or consequences** | A list of symptoms and consequences caused by the antipattern. |
| **Known exceptions** | Specific situations in which the antipattern may be acceptable. It is optional. |
| **Variations** | Optional item that lists the major variations of the antipattern. Additionally, alternative solutions should be described here. |
| **Related solutions** | List and citations cross-references suitable to the context of the antipattern. |

## 4. Correspondence Antipatterns

Due to possible precision errors that every ontology alignment tool is subject to, it may be the case that a correspondence included in an ontology alignment is not correct. Figure 1 exemplifies a fragment of the resulting alignment between two ontologies *o1* and *o2*[1]. Within each ontology, a rectangle around two classes indicates that these classes are disjoint, and a dotted line connecting two classes from different ontologies represents a correspondence.

Suppose that a terminology-based alignment tool identifies the following correspondences, represented in Figure 1:

$\langle$*o1:Document, o2:Document*, $\equiv$, 1.0$\rangle$ and

$\langle$*o1:Reviewer,o2:Review*, $\equiv$, 0.9$\rangle$,

where "$\equiv$" is the symbol for the equivalence relationship. Considering the commonly-known semantics of *"Reviewer"* and *"Review"* concepts, we intuitively know that the second correspondence is incorrect since it results in a logical contradiction, which is clear through the following argumentation: Suppose *x* is an instance of *o1:Reviewer*. Since *o1:Person* generalizes *o1:Reviewer*, *x* is also an instance of *o1:Person*. Considering the second correspondence, there is a possible

world *w* in which *x* is also an instance of *o2:Review*. Thus, *x* is also an instance of *o2:Document* in *w*, since it generalizes the *o2:Review* concept. Considering the first correspondence, *x* is also an instance of *o1:Document* in *w*. This results that *x* is an instance of *o1:Person* and *o1:Document* in *w* which, according to the disjoint relationship between *o1:Person* and *o1:Document*, is a logical contradiction [1]. In this work, we provide a way in which this intuition may be automatically inferred.

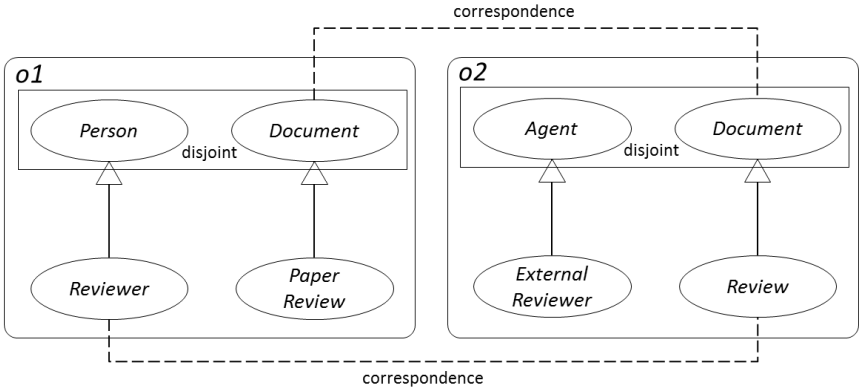


**Figure 1.** Fragment of two ontologies and an alignment problem. (Adapted from: [1]).

**Definition (Correspondence Antipattern):** Given two aligned ontologies *O* and *O'*, and their corresponding signatures *S* and *S´*, a correspondence antipattern *T* is a set of generic [2], domain-independent correspondences *D* and non-correspondences *N* between entities of *O* and *O´* that characterizes the existence of some problematic correspondence in *D*. Formally, it is defined as a tuple $T = \langle D, N, S, S' \rangle$.

An ontology correspondence antipattern *oca* is the result from a process that indicates an affirmation or a negation from a correspondence *D*, and may be represented as $oca = f(O, O', A, T, D)$. As a design (anti)pattern, *T* is specified as a template, that is, a theory referring to generic entities and their relations.

The purpose of a correspondence antipattern is to identify an incorrect correspondence *c* within an ontology alignment. This way, an instance *t* of a correspondence antipattern *T* occurs in an ontology alignment *A* between ontologies *O* and *O´*, in which $c \in A$. The search for an instance of *T* in a given ontology alignment *A* may be performed by exhaustively looking for all possible instantiations of *T* given *A*. The found instances represent evidences of this problematic solution.

For the development of correspondence antipatterns, the first step is to have the correct understanding of the problem being treated. When properly understood, the identified problem can result in correspondence antipatterns templates. Figure 2 presents our proposed methodology, which can assist in the construction of a correspondence antipattern. This methodology focuses on responding to key issues which are essential for an antipattern identification.

[2] Generic in the sense that it does not represent a specific concept only.

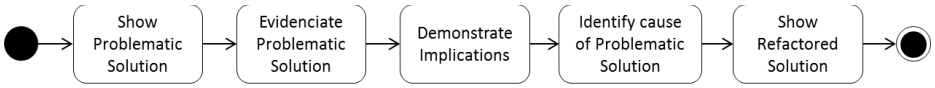


**Figure 2.**Methodology to construct a correspondence antipattern.

**Show Problematic Solution:** Correspondences, when incorrectly identified, may result in ontology artifacts that may be syntactically valid, but are likely to result in unintended domain representations. Thus, the set of valid cases (possible states of the domain of discourse) represented in the ontology artifact may not encompass the set of instances that do not represent desired states in this domain [11] or result in logical incompatibilities [1]. The first step to build an antipattern understands the problem that is being treated. What are the negative consequences that may result from using the bad solution (i.e., the set of matches)? In the scenario presented in Figure 1, we showed that the correspondence of the entities *o1:Reviewer* and *o2:Review* results in a logical contradiction according to the disjoint relationship between *o1:Person* and *o1:Document.* Then we have the following problem: *the correspondence of two subclasses whose superclasses are disjoint.*

**Evidenciate Problematic Solution:** for a solution to be considered problematic, this should in fact occur in practice, that is, there should be evidences that the bad solution is frequently applied. It is not reasonable to specify problematic solutions that do not even occur in practice. Following the example in Figure 1, an ontology alignment tool that only takes terminological similarity metrics into account will probably identify *o1:Reviewer* and *o2:Review* as correspondent classes, given the similarity of both strings.

**Demonstrate Implications:** For a correspondence antipattern specification to be complete and useful, it is important to discuss the reasons why an alignment solution is indeed problematic. In our previous example, given an instance *x* of *o1:Reviewer*, there is a possible world *w* in which *x* is also an instance of *o2:Review*, which in turn leads to the conclusion that *x* instantiates both *o1:Person* and *o2:Document* in *w*; however, *o1:Person* and *o2:Document* are disjoint, resulting in a logical contradiction. When the ontology alignment is being applied for the purpose of transforming instances or query reformulation [19], this will surely result in errors.

**Identify Cause of Problematic Solution:** The alignment fragment (that is, the subset of correspondences) that suffices to characterize it as a problematic solution should be identified. In the example being discussed, the two correspondences (between *o1:Reviewer* and *o2:Review*, and between *o1:Person* and *o2:Document)* cannot co-occur, due to the pre-defined disjoint relations. A domain-independent intuition for specifying this problematic solution is that there may never be two corresponding classes that are specializations of two disjoint classes. This correspondence antipattern (named as OCA01) can be formalized as:

OCA01: (?o1:?e1 ≡ ?o2:?e1) ⊓ (?o1:?e1 ⊑ ?o1:?e2) ⊓ (?o1:?e2 ⊓ ?o1:?e3 ⊑ ⊥) ⊓ (?o1:?e3 ≡ ?o2:e2) ⊓ (?o2:?e1 ⊑ ?o2:?e2).

For the construction and computational representation of a correspondence antipattern, we adopt EDOAL (Expressive Declarative Ontology Alignment Language), an open and agnostic language [12]. EDOAL is an extension of the alignment format proposed by Euzenat [24], and offers classes and relationships constructs, class restrictions, transformation of properties values, comparison operators for restriction on entities. The main advantages of EDOAL are that (i) it is independent from formalisms of ontological entities being aligned; (ii) it is an expressive model for

documenting correspondences and (iii) it complies with semantic web technology given its RDF and OWL syntax. EDOAL is used in [12] to represent correspondence patterns.

A fragment of the OCA01 correspondence antipattern is illustrated as follows, implemented using the EDOAL language:

```
<map>
   <cell>
      <entity1><Class rdf:about="?o1:?e1"/></entity1>
      <entity2><Class rdf:about="?o2:?e1"/></entity2>
      <relation rdf:resource="equivalence"/>
   </cell>
   <cell>
      <entity1><Class rdf:about="?o2:?e2" /></entity1>
      <entity2><Classrdf:about="?o2:?e1" /></entity2>
      <relation rdf:resource="subsumedBy"/>
   </cell>
   <cell>
      <entity1><Class rdf:about="?o1:?e1" /></entity1>
      <entity2><Class rdf:about="?o1:?e3" /></entity2>
      <relation rdf:resource="disjoint"/>
   </cell>
   <cell>
      <entity1><Class rdf:about="?o1:?e2" /></entity1>
      <entity2><Classrdf:about="?o2:?e2" /></entity2>
      <relation rdf:resource="equivalence"/>
   </cell>
   <cell>
      <entity1><Class rdf:about="?o1:?e2" /></entity1>
      <entity2><Class rdf:about="?o1:?e3" /></entity2>
      <relation rdf:resource="subsumedBy"/>
   </cell>
</map>
```

**Refactored Solution:** Refactoring in this case means repairing the alignment. In other words, when an instance of a correspondence antipattern is found in an alignment, the incorrect correspondence should be removed from the alignment.

An antipattern documentation template ensures that important issues are explicated for each antipattern. Table 2 lists the information that should be gathered during the development of a correspondence antipattern following the previous proposed methodology.

**Table 2.**Correspondence Antipattern OCA01

| Antipattern Item | Short Description |
|---|---|
| **Name** | OCA01 - Correspondence of two subclasses whose superclasses are disjoint. |
| **Refactored solution type** | Re-establishment of correspondence identified. |
| **Root cause** | Assume that two terms correspond to each other ignoring semantic relationship with other entities and their respective matches. |
| **Unbalanced forces** | A term mismatch may occur due to terminological homonyms, where similar terms with distinct meanings are claimed as corresponding entities. The semantics of the statement of terms must be observed for alignment composition. |
| **Antipattern general form** | $(?o1{:}?e1 \equiv ?o2{:}?e1) \sqcap (?o1{:}?e1 \sqsubseteq ?o1{:}?e2) \sqcap (?o1{:}?e2 \sqcap ?o1{:}?e3 \sqsubseteq \bot) \sqcap (?o1{:}?e3 \equiv ?o2{:}e2) \sqcap (?o2{:}?e1 \sqsubseteq ?o2{:}?e2)$. |
| **Symptoms and/or consequences** | Symptom: two subclasses that have equivalence correspondence, but their superclasses do not have equivalence correspondence. |
| **Known exceptions** | Not applicable. |
| **Variations** | Not applicable. |
| **Related solutions** | Not applicable. |

## 5. A Case Study on OAEI

The Ontology Alignment Evaluation Initiative (OAEI) is a coordinated international initiative whose goal is to evaluate the strengths and weaknesses of the ontology alignment tools, compare the performance and improve the assessment of ontology alignment techniques. Its main goal is to promote the continuous improvement of the ontology alignment tools. OAEI organizes annual campaigns addressing several domains, and publishes the results of the evaluated tools.

In this work, we manually scanned the published results from the 2013, 2012 and 2011.5 OAEI campaigns in the Anatomy, Benchmark, Conference, Library and Multi-Languages (called Multifarm) domains to extract all the reported incorrect correspondences resulted from the evaluated tools. We then ranked each correspondence from the reference alignments with regard to how many of the evaluated tools were not able to find this correspondence (that is, the most common errors). Due to space, this Section presents only two more correspondence antipatterns. The correspondences antipattern OCA02 and OCA03, documented below, occurred respectively 47 and 46 times in the OAEI published results we scanned. The correspondence antipattern OCA02 corresponds to the correspondence $\langle$*conference.paper, ekaw.paper,* $\equiv$, _$\rangle$ and the correspondence antipattern OCA03 corresponds to the correspondence $\langle$*confof.paper, ekaw.paper,* $\equiv$, _$\rangle$.

**OCA02 - Disjointness-subsumption contradiction with disjoint classes with subclasses.**

*Postulate:* Let e1, e2 and e3 be classes in an ontology o1, in which e2 is a subclass of e3, which in turn is disjoint with e1. If class e1 in ontology o1 equivalently corresponds to class e1 in ontology o2, class e2 in ontology o1 corresponds to class e2 in ontology o1 and class e2 in o2 is a subclass of e1 in ontology o2, then there is an alignment problem.

Table 3 shows a correspondence antipattern built from this problem, called OCA02.

**Table 3.**Correspondence Antipattern OCA02

| Antipattern Item | Short Description |
|---|---|
| **Name** | OCA02 - Disjointness-subsumption contradiction with disjoint classes with subclasses. |
| **Refactored solution type** | Re-establishment of correspondence identified. |
| **Root cause** | Assume that two terms correspond to each other ignoring semantic relationship with other entities and their respective matches. |
| **Unbalanced forces** | A term mismatch may occur due to terminological homonyms, where similar terms with distinct meanings are claimed as corresponding entities. The semantics of the statement of terms must be observed for alignment composition. |
| **Antipattern general form** | (?o1:?e1 ≡ ?o2:?e1) ⊓ (?o2:?e2 ⊑ ?o2:?e1) ⊓ (?o1:?e1 ⊓ ?o1:?e3 ⊑ ⊥) ⊓ (?o1:?e2 ≡ ?o2:?e2) ⊓ (?o1:?e2 ⊑ ?o1:?e3) |
| **Symptoms and/or consequences** | Symptom: two subclasses that have equivalence correspondence, but their superclasses do not have equivalence correspondence. |
| **Known exceptions** | Not applicable. |
| **Variations** | OCA01, OCA03 |
| **Related solutions** | Not applicable. |

**OCA03 - Disjointness-subsumption contradiction with disjoint classes without subclasses.**

*Postulate:* Let e1 be a class in ontology o1 that is disjoint with class e2 in the same ontology o1, and a class e1 in ontology o2 that specializes class e2 in o2. If class e1 in o1 equivalently corresponds to class e1 in o2 and class e2 in o1 equivalently corresponds to class e2 in o2, then there is a disjointness-subsumption contradiction alignment problem.

Table 4 shows a correspondence antipattern built from this problem, called OCA03.

**Table 4.**Correspondence Antipattern OCA03

| Antipattern Item | Short Description |
|---|---|
| **Name** | OCA03 - Disjointness-subsumption contradiction with disjoint classes without subclasses. |
| **Refactored solution type** | Re-establishment of correspondence identified. |
| **Root cause** | Assume that two terms correspond to each other ignoring semantic relationship with other entities and their respective matches. |
| **Unbalanced forces** | A term mismatch may occur due to terminological homonyms, where similar terms with distinct meanings are claimed as corresponding entities. The semantics of the statement of terms must be observed for alignment composition. |
| **Antipattern general form** | (?o1:?e1 ≡ ?o2:?e1) ⊓ (?o2:?e2 ⊑ ?o2:?e1) ⊓ (?o1:?e1 ⊓ ?o1:?e2 ⊑ ⊥) ⊓ (?o1:?e2 ≡ ?o2:?e2) |
| **Symptoms and/or consequences** | Symptom: two subclasses that have equivalence correspondence, but their superclasses do not have equivalence correspondence. |
| **Known exceptions** | Not applicable. |
| **Variations** | OCA01, OCA02 |
| **Related solutions** | Not applicable. |

## 6. Final Considerations

Ontology matching is a very active research field in the scientific community, where various techniques, approaches and tools have been proposed. However, such methods are still likely to identify incorrect correspondences between the entities of the ontologies that are being aligned. By identifying which errors frequently occur in the ontology matching process, it is possible to generalize such errors to specify correspondence antipatterns. Correspondence antipatterns assist in identifying incorrect correspondences or set of correspondences between ontologies that are being matched.

This work defined the concept of a correspondence antipattern and introduced a methodology to identify and computationally represent them. When analyzing a real alignment database, two incorrect solutions frequently found resulted in new correspondence antipatterns. The ontology alignment data used for this analysis was systematically extracted from OAEI, a representative initiative in the ontology matching field. We formally defined and proposed a methodology to assist the identification of correspondence antipatterns, as well as its documentation and computational representation. The language used to represent the correspondence antipatterns, EDOAL, is rich of constructs that allow the specification of complex

relations among the entities of the ontologies, and is also flexible to be extended given its XML, OWL and RDF syntax.

Further work will focus on the development of a reasoned to automatically recognize and refactor instances of a correspondence antipatterns on a given ontology alignment or set of alignments. Moreover, we are working on building a catalogue of correspondence antipatterns identified from all OAEI published results.

## References


[1] Meilicke, C.: Alignment Incoherence in Ontology Matching. PhD thesis Universitat Mannheim, Mannheim (2011)

[2] Scharffe, F., Zamazal, O. Fensel, D.: Ontology alignment design patterns. Knowledge and Information Systems, pp. 1–28 (2012)

[3] Brown, W., Malveau, R., McCormick, H., Mowbray, T.: AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis. John Wiley & Sons, New York (1998)

[4] Martens, A., Koziolek, H.: Performance-oriented Design Space Exploration, Components in a World of Mobile and Distributed Computing. Proc. 30th Int. Work. on Component-Oriented Programming (2008)

[5] Smith, C.U., Williams, L.G.> Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software. Addison-Wesley (2001)

[6] Navarro, E., Cuesta, C.E., Perry, D. E., González, P.: Antipatterns for architectural knowledge management. International Journal of Information Technology and Decision Making (2013)

[7] Ballis, D., Baruzzo, A., Comini, M.: A Minimalist Visual Notation for Design Patterns and Antipattern. Proc. 5th Int. Conf. on Information Technology: New Generations, IEEE Computer Society, Los Alamitos, pp.51-56 (2008)

[8] Stamelos, I.: Software project management antipatterns. Journal of Systems and Software 83(1) pp. 52-59 (2010)

[9] Presutti, V., Daga, E., Gangemi, A., Blomqvist, E.: eXtreme Design with Content Ontology Design Patterns. In: Proc. Workshop on Ontology Patterns, Washington D.C., USA (2009)

[10] Falbo, R., Barcellos, M., Nardi, J., Guizzardi, G.: Organizing Ontology Design Patterns as Ontology Pattern Languages. In: 10th Extended Semantic Web Conference (2013)

[11] Sales, T., Barcelos, F., Guizzardi, G.: Identification of Semantic Antipatterns in Ontology-Driven Conceptual Modeling via Visual Simulation. 4th International Workshop on Ontology-Driven Information Systems (2012)

[12] Scharffe, F.: Correspondence Patterns Representation. PhD thesis, University of Innsbruck (2009)

[13] Padilha, N.: Padrões e antipadrões de correspondências para melhoria do alinhamento de ontologias bem fundamentadas. Dissertação de Mestrado. Universidade Federal do Estado do Rio de Janeiro (2013)

[14] Guizzardi, G.: Ontological Foundations for Structural Conceptual Models, Ph.D. Thesis, University of Twente, The Netherlands (2005)

[15] Buschmann, F., Meunier, R., Rohnert, H., Sornmerlad, P.,Stal. M.: Pattern-Oriented Software Architecture. A System of Patterns, John'Wiley& Sons Ltd., England (1996)

[16] Fowler, M.: Refactoring. Reading, MA: Addison-Wesley (1999)

[17] Gruber, T.: Toward principles for the design of ontologies used for knowledge sharing. Int. J. Hum.-Comput. Stud.(1995)

[18] Sváb-Zamazal, O.: Pattern-based Ontology Matching and Ontology Alignment Evaluation, PhD Dissertation thesis, VSE-FIS, Prague(2010)

[19] Euzenat, J., Shvaiko, P.: Ontology Matching, Springer-Verlag Berlin Heidelberg (2007)

[20] Klein, M.: Combining and relating ontologies: an analysis of problems and solutions. In: Workshop on Ontologies and Information Sharing (2001)

[21] Predoiu L., Martin-Recuerda, F., Polleres, A., Feier, C. Mocan, A. de Bruijn, J. Porto, F. Foxvog, D. e Zimmermann, K.: Framework for representing ontology networks with mappings that deal with conflicting and complementary concept definitions. Technical report, DIP EU project, FP6 – 507483 (2004)

[22] Wache, H.:Semantisches mediation fürheterogeneinformationsquellen(2003)

[23] Ehrig, M.: Ontology Alignment: Bridging the Semantic Gap, Springer Science + Business Media, LLC (2007)

[24] Euzenat, J.: An API for ontology alignment. The Semantic Web–ISWC 2004, p. 698–712 (2004)
[25] Guarino, N.: Formal Onthology in Information Systems: Proceedings of the First International Conference (FIOS'98), June 6-8, Trento, Italy, vol. 46. IOS press (1998)
[26] Grau, B. C.,Dragisic,Z., Eckert, K., Euzenat, J., Ferrara,A., Granada, R.,Ivanova,V., Jiménez-Ruiz, E.,Kempf,A. O.,Lambrix,P.: Results of the Ontology Alignment Evaluation Initiative 2013, in *Proc. 8th ISWC workshop on ontology matching (OM)*, pp. 61–100 (2013)
[27] Jean-Mary, Y., Shironoshita, E., Kabuka, M.: Ontology matching with semantic verification. Journal Web Semantics: Science, Services and Agents on the World Wide Web archive. Volume 7 Issue 3, pp. 235-251 (2009)
[28] Guizzardi, G.: Ontological Foundations for Structural Conceptual Models, Ph.D. Thesis, University of Twente, The Netherlands.(2005)
[29] Guizzardi, G., Graças, A. P,Guizzardi, R. S. S.: Design Patterns and Inductive Modelling Rules to Support the Construction of Ontologically Well-Founded Conceptual Models in OntoUML", In: 3rd International Workshop on Ontology-Driven Information Systems (ODISE 2011), London, UK.(2011).