2009 14th IEEE International Conference on Engineering of Complex Computer Systems

Ontology Aided Model Comparison

Kleinner Oliveira [1], Karin Breitman[1]
Department of Informatics[1]
Pontifical Catholic University of Rio de Janeiro
Rio de janeiro, Brazil
{kfarias,karin}@inf.puc-rio.br

Toacy Oliveira[2]
DC School of Computer Science
University of Waterloo
Waterloo, ON, Canada
toliveira@csg.uwaterloo.ca

Abstract—**In this paper we explore the use of formal ontology in model comparison. Most techniques used in the MDA community are essentially syntactic, and rely on typographical hints, such as attribute data types and labels to determine similarity. We have adapted and incorporated an ontology alignment technique, as the means to obtain more precise and reliable similarity measurements between model elements, a fundamental issue in model comparison and composition.**

Keywords: **Ontology, UML, Model Comparison**

I. Introduction

UML models are universally used to describe real-life software system. They are useful to manage system descriptions, as they provide a set of models that capture different perspectives of the problem [1]. Quite frequently, during the software development process, practitioners need to compare and compose different UML models to provide desired solutions.

Model comparison requires a clear understanding of the UML metamodel specification and, of course, model semantics. Model composition relies on the ability to compare and indentify similarities and overlaps between model elements. Overlaps are undesired as they can lead to semantic conflicts, misinterpretation and problems in the model composition process. We define model composition as the process by which two input models, Ma and Mb, are combined producing model Mab as the result. In short, it can be represented by the expression: Ma (receiving) + Mb (merged) → Mab.

In the last few years, model comparison issues gained momentum at the software engineering community, bringing forth a variety of novel techniques, including schema matching [2], Web services composition, matching object catalogues, differences between versions of UML diagrams [5], and UML model comparison [6]. We have experimented with a few of the proposed approaches, but found none suitable for usage in UML model comparison. Such approaches ignore important aspects of model comparison that may lead to problems such as: (i) lack of flexibility to determine correspondences among model elements; (ii) lack of focus on model proprieties; (iii) require a large amount of human effort; (iv) do not take into account model meanings (the semantics). Thus, new approaches that overcome these shortcomings are in order.

In this paper we propose to capitalize from our previous experience in the project, implementation and integration of ontology based software applications [3], [6] to provide an enhanced solution to UML model comparison. We propose a match operator, responsible for putting in practice a strategy that takes into account both syntactic and semantic aspects involved during model comparison. Central to the functioning of the match operator is the capacity of finding precise similarity measurements between pairs of elements in different models in a flexible way. With this operator we are able to tackle and overcome most of the problems cited previously. We propose the use of an ontology-based match strategy in order to improve the calculation of element similarity, while preserving original model semantics.

The rest of this paper is organized as follows. In Section 2 we present our approach. In Section 3 we discuss how the similarity degree between two input models is calculated using the proposed approach. Finally, in Section 4, we present our concluding remarks and future work.

II. Model Comparison

Our model comparison approach, illustrated in Figure 1, focuses on the central process, in particular using ontology to enhance the Definition of Similarity Degree (S). On the first step of the comparison phase, the domain expert defines a signature for every model element type that can be defined as input model. On the second, he or she specifies a match strategy. The authors anticipate a choice among three pre defined and implemented strategies (i) *default*, (ii) *partial*, and (iii) *complete* match strategy (see [8], [9], [10] for more details). It is important to note that the approach is extensible and allows for the addition of new strategies, as a means of offering flexibility during the matching process.

This feature is particularly interesting if we take into account that matching strategies work better for some specific domains than others. This problem is very well known to the natural language processing community, where the application domain is a deciding factor in choosing matching strategies [12]. Once the match strategy has been determined, the match operator will put it in practice. Following is the stage in which the degree of similarity (S) between pairs of elements from the two different input models is calculated. Similarity is computed as the result of a combination of techniques. In this paper we extend the original architecture proposed in [8], [10] and include an ontology alignment strategy to enhance the results.

978-0-7695-3702-3/09 $25.00 © 2009 IEEE
DOI 10.1109/ICECCS.2009.55

The next step focuses on specifying equivalent model elements and producing the matching description. For this purpose, before comparing the models, the user sets a similarity threshold. Every pair of elements whose similarity degree is above the threshold is considered equivalent. The output consists of two sets of artifacts, the first is the match models, i.e., a set of models that are considered equivalent, and the second is a match description, i.e., a list of mappings between the input models. Figure 1 illustrates the process. In the next section we detail the techniques used in the computation of the similarity degree (computation of the similarity degree between pairs of input model elements using the ontology based strategy implemented by CATO), introducing the ontology based strategy.

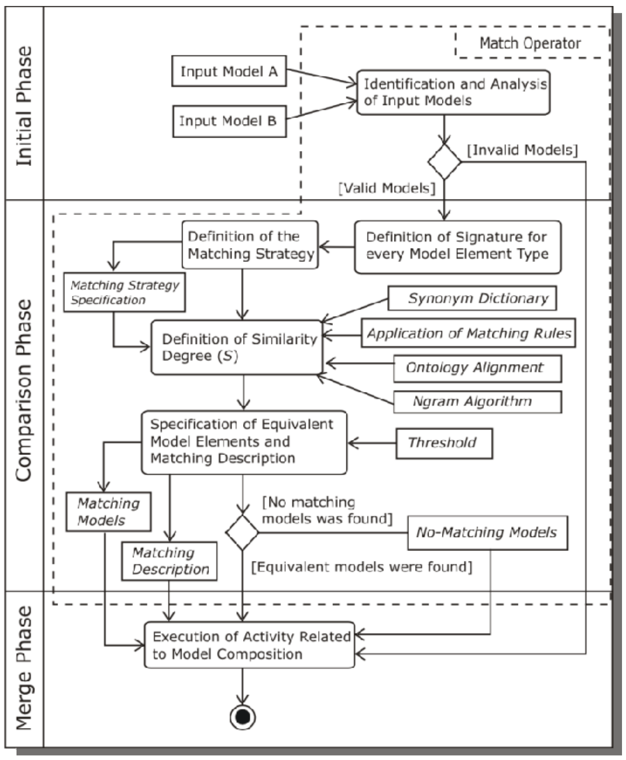
Initial Phase
Comparison Phase
Merge Phase
Match Operator
Input Model A
Input Model B
Identification and Analysis of Input Models
[Invalid Models]
[Valid Models]
Definition of the Matching Strategy
Definition of Signature for every Model Element Type
Matching Strategy Specification
Synonym Dictionary
Application of Matching Rules
Definition of Similarity Degree (S)
Ontology Alignment
Ngram Algorithm
Specification of Equivalent Model Elements and Matching Description
Threshold
Matching Models
[No matching models was found]
No-Matching Models
Matching Description
[Equivalent models were found]
Execution of Activity Related to Model Composition

Figure 1. Proposed model comparison process approach (adapted from [11]).

III. Similarity Degree Computation

To calculate the similarity degree between pairs of model elements, both syntactical and semantic aspects need to be taken into account. On the syntactical hand, both lexical and structural comparisons are performed in order to determine whether model elements in different input model should be considered syntactically equivalent. On the semantic hand, domain specialist expertise and ontologies are integrated in a seamless way. On a subsequent step, all strategies are then combined to determine the equivalence degree (S) between model elements. Synonym dictionary, typographic similarity, model signatures, verification using matching rules, and ontology-based strategy are underlying components to implement and put our approach in practice. They are described as follows.

A. Synonym Dictionary

With a synonym dictionary it is possible to identify mappings among domain concepts that have equal semantic value. The great benefit of using synonym dictionaries is to pave the way for the domain specialists to explicitly apply their domain expertise in the matching process. We denote by D(r,m) in [0,1] the degree of similarity between receiving (r) and merged (m) model elements. D(r,m) returns 0 whether r and m are synonym, otherwise it returns 1. D is calculated for every possible pair of (r,m).

Initially, every pair (r,m) of input model elements are not assumed to be synonymous, then D(r,m) = 0 for every pair of (r,m). In the current implementation of our model composition tool, called MoCoTo, the dictionary contemplates synonyms, hyponyms and hypernyms only. It is of particular interest to the model comparison process to include other semantic relationships as well, especially meronymical ones (part-of relationships) that although very frequently found in UML models, are very hard to identify automatically.

B. Typographic Similarity

The goal of typographic similarity is to determinate T(r,m) in [0..1] for every possible pair of receiving (r) and merged (m) model elements, Ma and Mb respectively. The N-gram algorithm [11] is applied to assign a similarity value in [0..1] to every possible pairs of (r,m). These pairs are determined by the cartesian product of (RxM), where R and M are the set of receiving and merged model elements, respectively. The result of RxM is a matrix. This algorithm yields a similarity degree to a pair of strings based on counting the number of their identical substrings of length N (we use N = 2).

C. Typographic Similarity

The signature is defined in terms of syntactic properties, where a syntactic property of a model element defines its structure. The signature is a collection of values assigned to a subset of syntactic properties in a model element's metamodel class. If an instance of a Class is an abstract class then isAbstract = true for the class, otherwise the instance is a concrete class, isAbstract = false. The set of syntactic properties used to determine a profile element's signature is called a signature type, as defined in [12].

We defined three types of signatures: (i) complete signature, which consists of all syntactic properties associated with a model element; (ii) partial signature, which is made up a range of syntactic properties; and (iii) default signature, which is composed only by properties name. The signatures can be structured in comparison levels organized hierarchically. Every model element type should have a signature. The similarity degree based on signature M between receiving (r) and merged (m) model elements is represented by M(r,m), where $0 \leq M \leq 1$. It is defined by calculating the weighted average among the arithmetic average of the levels (Equation 1):

$$\mathcal{M} = \frac{\sum_{i=1}^{n} p_i \cdot \left[\sum_{j=1}^{k} \frac{\varphi_{i,j}}{k} \right]}{\sum_{i=1}^{n} p_i} \rightarrow [0..1] \qquad (1)$$

- n is the number of levels employed to compare the model elements, where $n \geq 1$ and $n \in \mathbb{N}^*_+$. For example, we defined three levels to compare classes from input models. The first level has the property *name: String* only. The second one has the *ownedAttribute: Property*. The third one has the *ownedOperation: Operation*.
- p_i represents the weight, being $p_i = i$, where $i \geq 1$ and $i \in \mathbb{N}^*_+$; k expresses the number of elements in each level, where $k \geq 1$ and $k \in \mathbb{N}^*_+$;
- $\varphi_{i,j}$ (i and j represent the level and item of model elements that are being compared, respectively) is used to denote if an item in the receiving model element is equivalent to another item in the merged model element. It is a boolean variable and its value is determined by matching rules (described as follows). The matching rules compare pairs of items from model elements; returns 1 if the rule is satisfied, otherwise it returns 0.

D. *Verification Using Matching Rules*

In order to check if pair of input model elements is equivalent, we defined matching rules. The match operator is responsible to execute these matching rules and, according to the resulting of this execution, it defines consequently the value of $\varphi_{i,j}$, which was specified earlier. For every model element and item of model element, a matching rule to check if they are equivalent is necessary. This checking is based on the element's signatures. If a matching rule fails, then the models are not equivalent ($\varphi_{i,j} = 0$). Otherwise, models are equivalent ($\varphi_{i,j} = 1$).

The matching rules verify whether the input model element properties have the same values, and for each matching strategy is defined a set of matching rule according to respective signature type of the strategy. There are three kinds of matching rules: (i) *default matching rules* are a set of rules that compare models based on only their name, using the default signature type; (ii) *partial matching rules* are also a set of rules that compare models based on a number of syntactic properties of the models, using the partial signature type; (iii) *complete matching rules* are also a set of rules that compare models based on their syntactic properties, but use the complete signature type. Thus, the match operator makes use of these rules to implement the default, partial and complete match strategies, respectively.

E. *Ontology Based Strategy*

In this paper we adapt from an existing ontology integration technique strategy as an innovative means to obtain more precise similarity measurements. Before detailing the approach, however, we briefly argue in favor of the adoption of ontologies into the model comparison process.

1) *Why Ontology?*

Ontologies are much more expressive than other conceptual models: a controlled vocabulary is simply lists a set of terms and definitions, e.g. glossaries and acronyms; taxonomy is a set of terms arranged in a generalization-specialization (parent-child) hierarchy. A taxonomy may or may not define attributes of these terms nor does it specify other relationships between terms, e.g. RosettaNet and ebXML; a relational database schema defines a set of terms through classes, attributes and a limited set of relationships among those classes; an OO software model defines a set of concepts and terms through a hierarchy of classes and attributes and a broad set of binary relationships among classes. Constraints and other behavioral may be specified through methods on the classes (or objects). An ontology can express all of the preceding relationships, models and diagrams as well as, n-ary relations, a rich set of constraints, rules relevant to usage or related processes and other differentiators including negation and disjunction [13].

Furthermore ontologies capture knowledge rather than data. Because it is possible to infer new information from previously coded one (with the aid of an inference mechanism), we believe ontologies provide a much more robust conceptual model for model comparison in the MDA context, than restricting ourselves to pure UML models, that are neither formal nor support automated reasoning.

2) *Ontology-Based Mappings Semantic*

Mapping between two ontological models results in a formal representation that contains expressions that link concepts from one ontology to the second [14]. This result is of particular interest to the UML model comparison process approach proposed in this paper, for it provides formal, unambiguous, accurate and precise similarity measurements for pairs of model elements, while preserving their original semantics. A similarity measurement is represented by O, $0 \leq O \leq 1$ and $O \in \mathbb{R}_+$. Initially designed to provide mappings between two input ontologies, the ontology alignment strategy proposed in [15] is implemented by the CATO tool [16].

CATO takes as input any two ontologies written in W3C recommended standard OWL. It was fully implemented in JAVA and uses a specific API (Application Programming Interface) that deals with ontologies, JENA. It performs both lexical and structural comparisons in order to determine if concepts in different ontologies should be considered semantically equivalent. It is based in a refinement approach, broken into three successive steps, detailed in what follows. CATO's original output is an ontology that contains the elements from the input ontologies plus the mappings between those (if any). For the model comparison purposes described in this paper this result is of little interest, what is important is a byproduct of the ontology alignment process, the calculation of the similarity between pairs of elements from the two input ontologies. Figure 2 depicts CATO ontology alignment strategy.

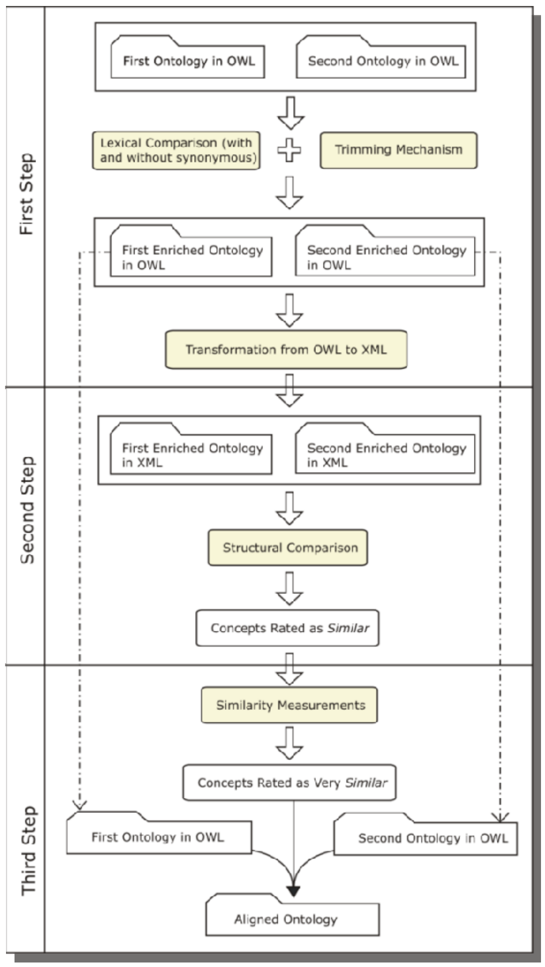
First Step
Second Step
Third Step
First Ontology in OWL
Second Ontology in OWL
Lexical Comparison (with and without synonymous)
Trimming Mechanism
First Enriched Ontology in OWL
Second Enriched Ontology in OWL
Transformation from OWL to XML
First Enriched Ontology in XML
Second Enriched Ontology in XML
Structural Comparison
Concepts Rated as Similar
Similarity Measurements
Concepts Rated as Very Similar
First Ontology in OWL
Second Ontology in OWL
Aligned Ontology

Figure 2. CATO ontology alignment strategy

3) First Step: Lexical Comparison

The goal of this step is to identify lexically equivalent concepts. We assume that lexically equivalent concepts are also semantically equivalent in the domain of discourse under consideration, an assumption which is not always warranted. Each concept label in the first ontology is compared to every concept label present in the second one, using lexical similarity as the criteria. Filters are used to normalize the labels to a canonical format: (i) if the concept is a noun, the canonical format is the singular masculine declination; (ii) if the concept they represent is a verb, the canonical format is its infinitive.

Besides using the label itself, synonyms are also used. The use of synonyms enriches the comparison phase because it provides more refined information. Lexical similarity alone, however, is not enough to assume that concepts are semantically compatible. We also investigate whether their ancestors share lexical similarity. It is important to note that the alignment strategy in this step is restricted to concepts and instances of the ontology. We are not considering properties at this time. A concept instance is represented by a pair name and namespace in OWL. As a result of the first stage of the proposed strategy, the original ontologies are enriched with links that relate concepts identified as lexically equivalent.

4) Second Step: Structural Comparison

Comparison at this stage is based on the subsumption relationship that holds among ontology concepts. Ontology properties and restrictions are not taken into consideration. Our approach is thus more restricted than the one proposed in [17], that analyses the ontologies as graphs, regarding both taxonomic and non taxonomic relationships among concepts. Because we only consider lexical and structural relationships in our analysis, we are able to make use of well-known tree comparison algorithms. We are currently using the TreeDiff implementation available at [18]. Our choice was based on its ability to identify structural similarities between trees in reasonable time. The goal of the TreeDiff algorithm is to identify the largest common substructure between trees, described using the DOM (Document Object Model) model. This algorithm was first proposed to help detect the steps, including renaming, removing and addition of tree nodes, necessary to migrate from one tree to another (both trees are the inputs to the algorithm).

The result of the Tree Diff algorithm is the detection of concept equivalence groups. They are represented as subtrees of the enriched ontologies. Concepts that belong to such groups are compared in order to identify if lexically equivalent pairs can also be identified among the ancestors and descendants of the original pair. Differently from the first step, where we based our analysis and compared concepts that were directly related to one another, we are now considering the structural vicinity of concepts. Every concept in the equivalence group is investigated in order to determine lexically equivalent pairs, number of matching sons, number of synonymous concepts in the sub-trees, available from the previous step, and ancestor equivalence.

5) Third Step: Fine Adjustments based on Similarity Measurements

The third and last step is based on similarity measurements. Concepts are rated as very similar or little similar based on pre-defined similarity thresholds. We only align concepts that were both classified as lexically equivalent in the second step, and thus rated very similar. Thus the similarity measurement is the deciding factor responsible for fine tuning our strategy. We adapted the similarity measurement strategies proposed in [18] (see Figure 3). Their similarity level is calculated in the present step. The final ontology will provides a common understanding of the semantics represented by the two input ontologies. As long-term goal, this representation can now be accessed by model comparison operator searching for information or knowledge to compare UML models. In this paper, we make use of the similarity measurements, represented by O, as cited previously and discard the final ontology and the element mappings.

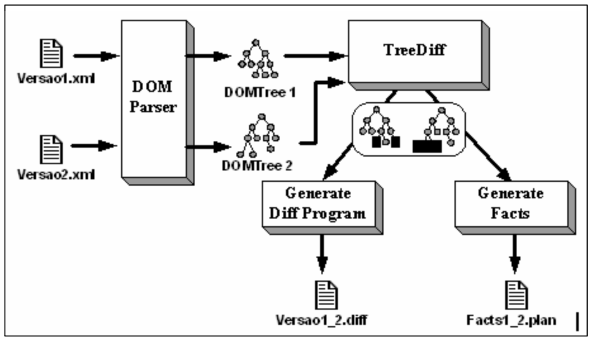

Figure 3. TreeDiff algorithm's entries and exits [18]

F. *Calculating the Similarity Degree between the Input Model Elements*

We denote by S the degree of similarity between receiving (r) and merged (m) model elements. For defining the similarity degree, it is necessary to combine the partial similarity degrees described in previous sections. For this purpose, it is calculated the average of D, T, M, and O as showed in Equation 2. If D = 1, then T also assumes value 1 and contrariwise.

$$S = \frac{D+T+M+O}{D+3} \rightarrow [0..1] \qquad (2)$$

Where:

D – Synonym Dictionary similarity degree, calculated as indicated in the previous section. Note that if D = 1, then T also assumes value 1. By the same token, if D=0, T assumes the same value.

T – Typographic Similarity, calculated as indicated in Section 4.2

M – Model Signature Similarity, calculated as indicated in Section 4.3

O – Ontology Alignment Similarity, calculated as indicated in Section 4.5

Based on the Equation 2, we calculate the similarity degree of every model elements. if S(r,m) > t, then r and m are equivalents, where t is a threshold and serves to determine whether pairs of model should be computed as equivalent. The possibility of combining different matching strategies assures overall better performance and reliability in the comparison phase. The ontology based approach is fully implemented and incorporated in the MoCoTo tool, as an Eclipse Plug-in which allows for a seamless integration with Eclipse Platform. It provides functionalities for users work with model composition and model comparison in the Eclipse SDK. The goal of the MoCoTo tool is to automatically compose pairs of input models.

IV. CONCLUDING REMARKS AND FUTURE WORK

In this paper we explored the use of ontology to enhance a model comparison mechanism based on similarity measurements. Most existing techniques are essentially syntactic in nature [12], [4], i.e., they make use of syntactical hints, such as attribute data types and naming similarities. It assumes that syntactical proximity implies semantic similarity, but such assumptions are often unwarranted and can lead to incorrect mappings [19]. We have adapted and incorporated an ontology alignment technique, as a means to obtain more precise and reliable similarity measurements between model elements, a fundamental issue in model comparison and composition.

We have only started experimenting with the ontology based approach, but the results already show improvement from the ones obtained using syntactic based techniques alone. Further empirical studies are necessary, however, to calibrate the similarity thresholds used in the ontology based approach, validate the approach in real world design settings, and verify its performance levels and its applicability in different application domains. Obviously scalability is an issue, and more investigation on the applicability of the proposed approach in comparing large UML models is required.

REFERENCES

[1] S. Sendall and W. Kozaczynski, "Model Transformation: The Heart and Soul of Model-Driven Software Development," IEEE Software, vol. 20, no. 5, September/Octocber 2003, pp. 42–45.

[2] E. Rahm and P. Bernstein, "A Survey of Approaches to Automatic Schema Matching," Very Large Data Bases Journal, vol. 10, no. 4, pp. 334-350, 2001.

[3] L. Leme, D. Brauner, K. Breitman, M. Casanova, and A. Gazola, Matching Object Cataloques, Inovations in System Software Engineering, Springer, 2008.

[4] D. Ohst, M. Welle, and U. Kelter, "Differences between Versions of UML Diagrams," Proc. 9th European Software Engineering Conference, ACM Press, pp. 227–236, 2003.

[5] D. Kolovos, R. Paige, and F. Polack, "Model Comparison: a Foundation for Model Composition and Model Transformation Testing," Proc. International Workshop on Global Integrated Model Management, New York, NY, USA: ACM Press, pp. 13–20, 2006.

[6] K. Breitman, M. Casanova, and W. Truszkowski, Semantic Web: Concepts, Technologies and Applications, Springer Verlag, 2007.

[7] Unified Modeling Language: Infrastructure version 2.1, OMG, February 2007.

[8] K. Oliveira, "Composição of UML Profiles," Master's thesis, Informatics Faculty, Pontifical Catholic University of Rio Grande do Sul, Porto Alegre, Brazil, February 2008.

[9] K. Oliveira and T. Oliveira, "A Guidance for Model Composition," Proc. International Conference on Software Engineering Advances (ICSEA'07), pp. 27–32, August, 2007.

[10] K. Oliveira and T. Oliveira, "Model Comparison – A Strategy-Based Approach," Proc. 20th International Conference on Software Engineering and Knowledge Engineering, San Francisco, USA, 2008.

[11] C. Manning, and H. Schutze, Foundations of Statistical Natural Language Processing, ISBN 978-0262133609, MIT Press, 1999.

[12] Y. Reddy, R. France, G. Straw, N. Bieman, E. Song, and G. Georg, “Directives for Composing Aspect-Oriented Design Class Models,” Transaction on Aspect-Oriented Software Development (AOSD), vol. 1, no. 1, pp. 75–105, 2006.

[13] A. Pérez, M. Peréz, and O. Corcho, Ontological Engineering, Springer Verlag, 2004.

[14] J. Euzenat and P. Shvaiko, Ontology matching, Springer, Springer-Verlag, Berlin Heidelberg (DE), 2007.

[15] C. Felicissimo, “Interoperabilidade Semântica na Web: Uma Estratégia para o Alinhamento Taxonômico de Ontologias,” Master’s thesis, Department of Informatics, Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, Brazil, August 2004.

[16] K. Breitman, C. Felicissimo, and M. Casanova, “CATO - A Lightweight Ontology Alignment Tool,” Proc. International Conference on Advanced information Systems Engineering (CAiSE), Short Paper Proceedings, 2005.

[17] F. Noy and A. Musen, “The PROMPT Suite: Interactive Tools For Ontology Merging And Mapping,” International Journal of Human-Computer Studies, 2003.

[18] U. Bergmann, Evolução de Cenários Através de um Mecanismo de Rastreamento Baseado em Transformações, PhD Thesis of the Department of Informatics of PUC-Rio, 2002.

[19] M. Casanova, K. Breitman, F. Brauner, and A. Marins, Database Conceptual Schema Matching. Computer (Long Beach), v. 40, p. 102-104, 2007.