Semantic Interoperability by Aligning Ontologies*

Karin Koogan Breitman[1], Carolina Howard Felicíssimo[1], Luiz Marcio Cysneiros[2]

[1] Departamento de Informática - PUC-Rio
{karin, cfelicissimo}@inf.puc-rio.br

[2] York University
{cysneiro}@ mathstat.yorku.ca

Abstract. A fundamental premise of the semantic web is that a level of interoperability is guaranteed among applications running in an open environment. By the use of ontologies applications are able to share information and exchange meaningful data. In this context ontology alignment is paramount to assure communication among applications. In this paper we propose an ontology alignment strategy based in the concepts of partial alignment and inconsistency tolerance.

1. Introduction

Researchers from industry and academia are now exploring the possibility of creating a "Semantic Web," in which meaning is made explicit, allowing machines to process and integrate Web resources intelligently. This technology will allow interoperability among development of intelligent internet agents in large scale, facilitating communication between a multitude of heterogeneous web-accessible devices. Unfortunately the majority of the information available is in a format understandable to humans alone, thus creating the need to provide an adequate amount of semantics to allow for some of the information filtering to be done by machines. The emergent technology to address this problem is the codification of the information using ontologies, i.e., conceptual models that embody shared conceptualizations of a given domain [11].

We believe the task of ontology building belongs to requirements engineers - after all we are trained in conceptual modeling techniques, and that is what building ontologies is all about. We have developed a process in which we use a special lexicon as a starting point to building ontologies [26, 25]. The idea of using a glossary of terms in the early development of ontologies is not new, and it is supported by some ontology development methodologies [22, 10, 12]. The basic idea is to start with an informal definition and use a stepwise refinement process until the desirable

* This research was supported in part by CNPq under contract ESSMA- 552068/2002-0, by Faperj under Student Grade 10 scholarship and by CAPES.

level of formality is achieved. In our case the output of the process is a machine processable ontology written in DAML + OIL.

We are currently developing automated support to our process using the open source C&L tool [8], available at http://sl.les.inf.puc-rio.br/cel/aplicacao/. The tool was initially proposed as a lexicon and scenario edition and management environment. The ontology generation plug-in is currently under way and is scheduled to be made public in October, 2003. In Fig. 1 we depict the lexicon based ontology construction process.

Central to our research is Tim Berner's Lee [2] belief that in the near future every site and web application will have to make available its ontology and Jim Hendler's notion that instead of having a few carefully crafted (by AI experts, such as the WordNet [9] and CYC efforts [13]) ontologies, there will be a "*great number of small ontological components consisting largely of pointers to each other*" [14]. The result will be a great variety of lightweight ontologies both built and maintained by independent parties (not necessarily with expertise in ontology development).

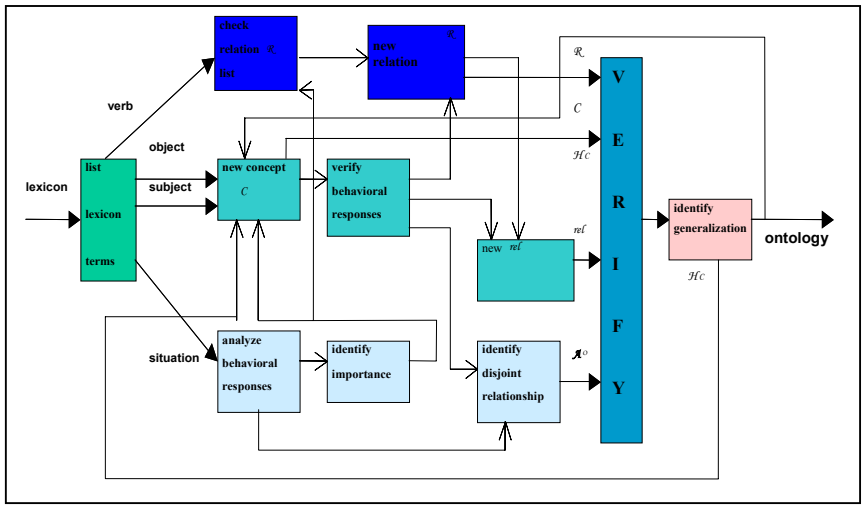

Fig. 1. Lexicon Based Ontology Construction Process

Our recent experiences with ontologies have demonstrated that ontology development is not particularly challenging compared to building any other conceptual model used in our RE practice such as KAOS [3] and i* [23]. The constructs are not extremely complex, neither is the level of formality required. Any person familiar with basic concepts of first order logic and the notions of *subsumption* and *aggregation*, should not experience major difficulties in the process. In terms of implementation languages, although today's scenario may seem confused, there seems be a convergence to DAML+Oil (or its equivalent OWL DL sublanguage) and a consensus

that the language while being expressive enough, still allows for adequate automatic support, as put by the W3 consortium "*maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computed) and decidability (all computations will finish in finite time).*" In addition there are editors, such as OilEd, that provide automated support to the edition and maintenance of ontologies. Structural consistency can be automatically verified with the FaCT tool. Both tools are available at http://oiled.man.ac.uk/.

2. Ontology Alignment

The real bottleneck is, in our opinion, to secure what is commonly referred to as *"semantic interoperability"*. That means that open system applications with different ontologies will have to undergo a negotiation process. This operation is named ontology alignment and it aims at an intermediate representation that can be shared by both applications. While aligning ontologies one can merge the two ontologies into one, integrate (negotiate and decide on a representation that uses concepts from both ontologies) or simply translate.

Available tools such as Chimaera [18] implement ontology integration algorithms that provide the full mapping between two different ontologies. This process is time consuming and requires a great amount of user intervention. Our perception is that there is little chance of success in trying to obtain total alignment between ontologies. The effort involved is too great and may not be justifiable in the context in which the ontologies will operate. A salient problem is the duration of interaction between two applications - do we have enough time to align the ontologies?

In order to make semantic interoperability possible in the context of the semantic web it is paramount to devise an ontology integration process that is fast (real time, ideally), reliable and does not rely on user intervention. The last because we can not assume that either users or agents will have enough domain knowledge as to provide useful input in the integration process.

The ontology integration algorithms available, such as Prompt and Glue, lead to a long and computationally expensive process [19, 5]. Our perception is that in most interactions between ontology based agents a full integration may not be needed, rendering most part of the integration effort useless. In this light we propose a new approach to ontology integration. Our idea is to partially integrate ontologies, mapping ontological primitives (concepts, properties, axioms, restrictions and comparing concept trees), as they are needed. In the next section we describe our proposal.

3. Partial Ontology Integration

In order to make partial alignments one must accept the presence of inconsistency. In software engineering, it has long been recognized that inconsistency is a fact of life. Evolving descriptions of software artifacts are frequently inconsistent, and tolerating

this inconsistency is important if flexible collaborative working is to be supported [7]. We believe that the same principles can be applicable in the case of aligning ontologies. Assuming that inconsistencies are bound to arise, one may choose from one of two approaches. In an early binding approach the two ontologies will be compared and all inconsistent parts will be discarded. This approach is not desirable because it may leave outside concepts that may be fundamental to a given transaction. The alternative is a late binding approach, that focuses in dealing with inconsistencies as they are found. Once an inconsistency is detected, one must decide what action is to be taken. Nuseibeh et al identified three possible courses of action in the presence of inconsistency: ignore, tolerate and resolve [20].

We divide our strategy in two steps: inconsistency detection and action taking.

3.1 Ontology Detection

The initial requirement to align different ontologies is being able to list differences and inconsistencies between both ontologies. We are currently specifying a mechanism to detect such differences between ontologies. In particular we are aiming at identifying:

Concepts using different names (labels) for the same meaning:

- Differences in the number of restrictions (differentiate among cases where there is intersection of restrictions);
- Differences in the properties used in the restriction - related concepts are similar;
- Differences in the related concepts used in the restriction - properties are similar.

Concepts with the same name (label) with different meaning:

- Identify differences in restrictions;
- Identify differences in the properties used.

Properties with different name (label) and same meaning:

- Verify if all concepts the properties relate are equivalent in both ontologies.

Properties with the same name (label) and different meaning:

- Verify if the concepts that use the property are consistent in both ontologies.

Early attempts were made comparing the ontologies in their native representation language, DAML+OIL. Some problems arose from the expressiveness (or lack) of DAML+Oil itself. Neither concepts nor properties accept synonyms, therefore the concept *dog* from *Ontology1* and *dogs* from *Ontology2* would be considered different. The use of synonyms could alone avoid mismatches caused by plural/singular (*dog*/*dogs*), male/female (*salesman*, *saleswoman*) and verbal time (*pays*, *pay*).

We are currently experimenting with the intermediate version we produce while applying our process to the lexicon. In this representation we have a database that contains information present in the lexicon and the ontological structure that is built as a result of the application of the process. This repository contains more information that is currently provided by DAML+OIL, synonyms and structured descriptions of the terms of the lexicon (denotation and connotation). At this point, we have not experimented enough to make any suggestions to a possible need of additional information to the DAML+OIL notation. We have noticed, however, that the use of synonyms decreased the number of items in the list of discrepancies. Of course this attempt can only be applied to ontologies to which we have a lexicon available, more so, a lexicon modeled using the LEL notation [26]. An alternative is to rely on the use of external, all purpose thesauri.

3.2 Action

Once a list of inconsistencies is detected we must decide what action is to be taken. As mentioned before three courses of action are possible: ignore, tolerate and resolve [20]. We are going to use similarity measurements to aid the decision making process. We are currently reviewing and adapting similarity identification techniques that we used in the past, in the context of scenario based software development [4, 1]. The similarity measurements used in the scenario context take into consideration the primitive scenario elements, e.g., title, resources, actors. In the case of ontologies we are considering the primitives as defined by Maedche in [17]. According to the author, an ontology can be described by a 5-tuple consisting of the core elements of an ontology, i.e., concepts, relations, hierarchy, a function that relates concepts non-taxonomically and a set of axioms. The elements are defined as follows:

O: = {C, R, H^C, rel, A^O} consisting of:

- Two disjoint sets, C (**concepts**) and R (**relations**);
- A **concept hierarchy**, H^C: H^C is a directed relation $H^C \subseteq C \times C$ which is called concept hierarchy or taxonomy. H^C (C1, C2) means C_1 is a subconcept of C_2;
- A **functio**n rel: $R \rightarrow C \times C$ that relates the concepts non taxonomically;
- A set of ontology **axioms** A^O, expressed in appropriate logical language.

For the sake of similarity measurements we are going to be considering the concepts, relations, the *rel* function and axioms as our primitives. The similarity between hierarchies will measured using a different strategy, based on the comparison of abstract data trees. We are currently experimenting with the TreeDiff algorithm proposed in [24] that identifies the largest approximately common substructure of two trees. This algorithm also will may helps in the detection of inconsistencies between ontologies.

Scenarios are artifacts used in the stages of requirements elicitation and modeling to help identify relevant situations in the domain. Similarly to ontologies, scenarios are written in natural language and, as opposed to being represented as a block, are

composed of independent elements [16]. A typical scenario contains a goal, context, a description of the resources available, the actors involved and episodes [16], whereas an ontology has two disjoint sets, concepts and relationships, a hierarchy, a function that relates the concepts and uses relationships (restriction) and, of course a list of axioms [17].

Because of the hierarchical structure of ontologies they are suited to the application of similarity detection strategies based on database vision integration mechanisms, that aim at the identification of structural relationships among entities, aggregation, composition and such. The application of a similarity detection technique can point to one of four possible results, depicted in Fig. 2 as follows.

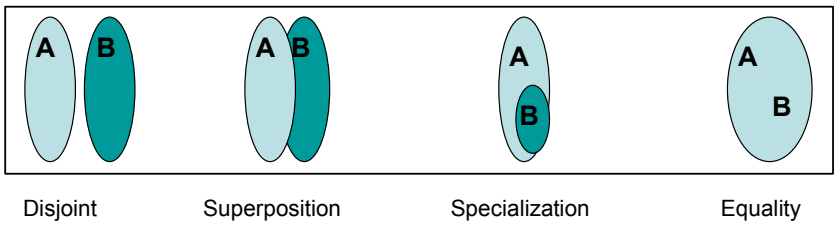

Fig. 2. Similarity degree between entities A and B

The similarities thresholds, i.e., how much is necessary to qualify as enough to ignore, tolerate or resolve are not yet clear, and will have to be empirically determined. This experimentation will take place as soon as the similarity detection techniques are adapted and running for ontology comparison.

Once an inconsistency that requires resolution is encountered we will apply classical mapping solution, such as the ones implied by [19] and proposed by [15]. The greatest contribution of this approach is reducing the solution space of the number of concepts that actually need alignment, considerably. Firstly we decided to do partial alignment, thus leaving a number of concepts outside (accepting the inconsistencies), then, from the subset of concepts that are involved, we classify their inconsistencies in ignorable, tolerable and need resolution. Of the three, only the last will actually need alignment. We believe that this approach is more likely to succeed in the fast paced context of the semantic web and its demand for real time alignment in order to secure agent interoperability.

In addition, we are going to anchor our approach in the concept of a "preferred" ontology. We are assuming that in the context of the semantic web there will be some ontologies that are more carefully crafted, therefore more reliable, than others. Some domains, such as medical and finance systems, provide upper ontologies in an effort to standardize a common vocabulary of concepts and terms. Such ontologies are validated by a large community of users and official entities, and should be preferred

over unknown ontologies in the case of inconsistency. Such ontologies, could be recognized by the use of simple mechanisms, such as digital signatures and a list of reliable sources, and be given priority. In this case, the inconsistency shall be resolved by the mapping of the inconsistent concept to the preferred ontology. That could be implemented by a wrapper, for instance. Note that the preferred ontology may not even be among those ontologies involved in the alignment process, but could be provided by a third party, for instance.

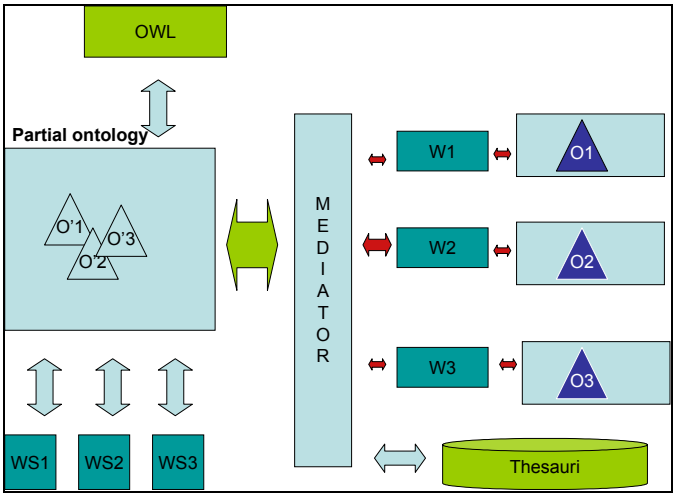

Fig. 3. Tentative architecture for the partial ontology integration approach

Finally, the use of thesauri has proven very useful in the scenario similarity detection process. This fact is due to the presence of synonyms in every known natural language. What could be detected as a difference may in fact be the misunderstanding of equivalent terms. The use of a thesaurus (or dictionary, lexicon, glossary) greatly contributed in eradicating such cases. We are going to incorporate the use of thesauri to our ontology alignment process. In Fig.3 we show a tentative architecture for our approach. The idea of the mediator is to accumulate the selected concepts (the ones whose inconsistencies will be tolerated plus the ones that have been aligned) and produce the resulting ontology. Once again, this ontology will be used at the requested interaction only and will be further discarded. We see no reason in making such artifact persistent, for the ontologies will evolve and chances are that, even in a much similar situation, other concepts shall be involved (and render this alignment useless).

4. Conclusion

We are convinced that the solutions to the ontology integration problem are intertwined with our abilities to tolerate and live with inconsistencies, that as put by Easterbrook and Chechnik are "a fact of live" [7]. It is not an easy shift however, for we have been trained to strive for completeness, consistency and to avoid conflict. We propose an architecture that takes that into consideration and allows for ontology integration in the presence of inconsistency. We are currently elaborating a prototype to help us adjust our similarity threshold measurements more empirically. We understand that measurements will be necessary in order obtain the ideal tuning of similarity measures. The last have to, ideally, allow for automatic alignment and yet retain a good level of reliability in the mappings made.

In parallel, we are investigating the possibility of a mechanism that translates the ontologies to a lexical representation. One of the reasons is to facilitate validation with users. We have noticed that the visualization of ontologies is somewhat difficult to users[1]. No tool, to the best of our knowledge, is able to display a broad overview of an ontology but, instead, most tools provide a fish eye view, concept per concept (as it is the case with OilEd [27] or Protègè-2000 [28]). Of course that in the conversion process, some information will be lost, for lexicons are flat, as opposed to the hierarchical structure of ontologies. There is also no clear way to represent an axiom in the lexicon.

In this context a few questions arise. Are the requirements for the interactions (in the semantic web) so ephemeral in nature that it is cost effective to allow the interaction even in the presence of inconsistency? How much mismatch/inconsistency is allowable? Are levels of similarity an acceptable measure? Is it possible to analyze the impact and the risks involved in tolerating inconsistency between ontological representations? Can we apply classical inconsistency handling approaches to ontologies, such as the one proposed in [20]?

5. References

1. Alspaugh, T.A.; Antón, A.; Barnes, T.; Mott, B. - An integrated scenario management strategy - Proceedings of the 4th. IEEE Symposium on Requirements Engineering (RE99), Limerick, Ireland 1999 - pp. 142-149.
2. Berners-Lee, T.; Lassila, O. Hendler, J. – The Semantic Web – Scientific American – May 2001 - http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html
3. P. Bertrand, R. Darimont, E. Delor, P. Massonet, A. van Lamsweerde GRAIL/KAOS: an environment for goal drivent requirements engineering - Proceedings ICSE'98 - 20th International Conference on Software Engineering, IEEE-ACM, Kyoto, April 1998.

1 Which, perhaps, is intentional, as ontologies for the semantic web and, in particular DAML+OIL, were created to provide MACHINE interoperability as opposed to facilitate human understanding.

4. Breitman, K.K.; Leite, J.C.S.P. – A framework for scenario evolution – in Proceedings of the Third International Conference on Requirements Engineering (ICRE) - Colorado Springs, USA– 1998. – pp. 214-221.
5. Doan, J. Madhavan, P. Domingos, and A. Halevy. - Learning to Map between Ontologies on the Semantic Web *WWW-2002*.
6. Easterbrook, S.; Chechik, M. - 2nd International Workshop on Living with Inconsistency - Summary, - IEEE - 2001.
7. Easterbrook, S.; Chechnik, M. - Introduction to the Second Workshop on Living with Inconsistency - Software Engineering Notes, Vol 26 No. 6 - ACM Press - 2001 - pp.76-78.
8. Felicissimo, C. H., Silva, L. F., Breitman, K. K., Leite, J. C. S. P. - Geração de Ontologias subsidiada pela Engenharia de Requisitos – VI Workshop on Requirements Engineering, Piracicaba, São Paulo, Brazil, November 2003.
9. Fellbaum, C.; ed - WordNet: An electronic Lexical Database - Cambridge, MA - MIT Press - 1998.
10. M. Fernandez, A. Gomez-Perez, and N. Juristo. METHONTOLOGY: From Ontological Arts Towards Ontological Engineering. In Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering, Stanford, USA, pages 33--40, March 1997.
11. Gruber, T.R. – A translation approach to portable ontology specifications – Knowledge Acquisition – 5: 199-220
12. Gruninger, M.; Fox, M. – Methodology for the Design and Evaluation of Ontologies: Proceedings of the Workshop on basic Ontological Issues in Knowledge Sharing – IJCAI-95, Montreal, Canada, 1995.
13. Guha, R. V., D. B. Lenat, K. Pittman, D. Pratt, and M. Shepherd. "Cyc: A Midterm Report." *Communications of the ACM* Vol.33 , No. 8 - August, 1990.
14. Hendler, J. – Agents and the Semantic Web – IEEE Intelligent Systems – March/April - 2001. pp.30-37
15. Klein, M.; Ding, Y.; Fensel, D.; Omelayenko, B.- Ontology Management: Storing, aligning and maitaining ontololgies - in Towards the Semantic Web: Ontology Driven Knowledge Management - Editors: John Davies, Dieter Fensel, Frank van Harmelen - Wiley 2003.
16. Leite, J.C.S.P. et al. – Enhancing a Requirements Baseline with Scenarios - – Requirements Engineering Journal vol(2) pp. 184-198 – Springer Verlag - December, 1998.
17. Maedche, A. – Ontology Learning for the Sematic Web – Kluwer Academic Publishers – 2002.
18. D. McGuinness, R. Fikes, J. Rice, and S. Wilder - *The Chimaera Ontology Environment*. In Proceedings of the 17th National Conference on Artificial Intelligence (AAAI), 2000
19. Noy, N. ; M.; Musen, M. - Prompt Algorithm and Tool for Automated Ontology Merging and Alignment - Proc. 17th Natl. Conf. on Artificial Intelligence (AAAI'2000), Austin, TX, July/August 2000
20. Nuseibeh, B.; Easterbrook, S.; Russo, A. - Leveraging Inconsistency in Software Development - Computer Vol 33 No. 4 - IEEE Computer Society Press - 2000 pp. 26-29
21. Nuseibeh, B.; Easterbrook, S.; Russo, A. - Leverage Inconsistency in Software Development - Computer - Vol 33 No. 4 - April 2000 - pp. 24-29
22. Ushold, M.; Gruninger, M. – Ontologies: Principles, Methods and Applications. Knowledge Engineering Review, Vol. 11 No. 2 – 1996. pp. 93-136
23. Yu, E. - Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering - Proceedings of the Third International Symposium on Requirements Engineering - RE97 - IEEE Computer Society Press - p1997 - pp.226-235
24. Wang, J. et al - an Algorithm for Finding the Largest Approximately Common Substructures of Two Trees, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 20, No. 8, 1998 - pp.889-895.

25. Breitman, K.K.,; Leite, J.C.S.P.- Lexicon Based Ontology Construction - 2nd. International Workshop on Software Engineering for Large Scale Multi Agent Systems - SELMAS - ACM computer Press, Portland Oregon, 2003.
26. Leite, J.C.S.P. and Franco, A.P.M. A Strategy for Conceptual Model Acquisition. First International Symposium on Requirements Engineering. Proceedings**.** IEEE Computer Society Press, 1993. pp. 243-246.
27. Disponível em <http://oiled.man.ac.uk/>. Acesso em 17 de Setembro de 2003.
28. Disponível em <http://protege.stanford.edu/>. Acesso em 17 de Setembro de 2003.