# Semi-Automated Consensus Finding for Meaning Negotiation

Christophe Debruyne, Johannes Peeters, and Allal Zakaria Arrassi

Semantics Technology and Applications Laboratory (STARLab)
Department of Computer Science
Vrije Universiteit Brussel
Pleinlaan 2, B-1050 BRUSSELS 5, Belgium
{chrdebru, johpeete, aarrassi}@vub.ac.be

**Abstract.** Finding a consensus between communities to create an ontology is a difficult task. An evolutionary process where domain experts and knowledge engineers work together intensively is needed to support collaborative communities in defining a common ontology. These communities model their view of a particular concept while knowledge engineers try to find a consensus. Negotiation, finding similarities and defining new points of interests are important processes to achieve such a consensus. To aid these processes we present several algorithms, built upon a state-of-the-art community grounded ontology evolution methodology. These algorithms are illustrated with an example.

## 1 Introduction

Communities share interests and information in order for them to collaborate. For such collaborations to be successful, they need instruments to achieve an agreement on how to communicate about those interests and information. One of these instruments is an ontology, which is a specification of a shared conceptualization [1]. The term quickly became popular in the field of knowledge engineering, where such conceptualizations are externalized formally in a computer resource. Externalization enables sharing and interoperation between information systems, the so called shared agreement [2].

When designing ontologies, we only want to conceptualize a relevant subset of the world as efficiently as possible [3]. An ontology engineering methodology is a vital instrument in order for collaborations to succeed. Such a methodology helps designing these formal, agreed and shared conceptualizations by making them reusable, reliable, shareable, portable and interoperable. It can also act as a basis for ontology engineering during class.

Starting from a common ontology, such a methodology requires the different communities to render their view of a new concept on the common ontology. This results in a divergence, which needs to be converged in order for a new version of the common ontology, containing an agreed view of that new concept, to appear. Ontology integration, of which a survey is provided in [4], takes care of that convergence. However, finding an agreement is a tedious task and requires all communities to work together and discuss their progress. These negotiation processes were discussed in approaches such as [5, 6]. We name this process *meaning negotiation*.

## 2 Problem Statement

One of the biggest challenges of meaning negotiation, after receiving all the views, is to define ways to guide that process. Meaning negotiation can be a semi-automated process, where different algorithms help signalling problems or streamline the results for future iterations. When problems arise (e.g. a different semantics for a certain concept), the different communities need to be contacted for clarity, discussion or negotiation.

Questions can be formulated to indicate where problems might rise. The knowledge engineer can answer these questions in order to construct an agenda for the meaning negotiation session. The questions that need to be answered are threefold:

1. Which relations did the different communities specialize or genereralize, by using more specialized or general concepts from a taxonomy?
2. Which are the stable parts in the ontology, type hierarchy or template? A stable part is a subset of concepts and their interrelationships, which were not changed by the different communities.
3. Which new concepts have to be elaborated in more detail, by defining relations around them, in a future iteration?

In this paper we propose a solution which answers the three questions mentioned above. We formulate the solution and present results from a realistic example in the following sections. We conclude with our findings and a discussion about some future directions, which we find worthwhile investigating.

## 3 Approach

In this section we present several algorithms answering the questions formulated in previous section. These algorithms are applied to the different views before the meaning negotiation session starts. The collaborative ontology engineering methodology that we extended is the DOGMA-MESS [3] ontology engineering methodology. We choose DOGMA-MESS for its explicit use of templates, which guide domain experts to give their conceptualization, not found in other ontology engineering methodologies such as HCOME [5] and DELIGENT [8].

### 3.1 DOGMA and DOGMA-MESS

The DOGMA[1] approach developed at STARLab aims at the development of a useful and scalable ontology engineering approach. In the DOGMA ontology

[1] DOGMA: Developing Ontology-Grouned Methods and Applications

engineering approach, ontology construction starts from a (possibly very large) uninterpreted base of elementary fact types called lexons [9, 10] that are mined from linguistic descriptions such as existing schemas, a text corpus or formulated sentences by domain experts.

A lexon is an ordered 5-tuple of the form $\langle\gamma, t_1, r_1, r_2, t_2\rangle$, where: $t_1$ and $t_2$ are respectively the *head-term* and *tail-term*, $r_1$ and $r_2$ the role and co-role and $\gamma$ is the *context* in which the lexon holds. A lexon can be read in both directions since the head-term plays the role in the relation and the tail-term the co-role. The context is used to disambiguate the terms in case of homonyms, e.g.: a capital in a geographical context is not the same as a capital in an economical context. An example of a lexon is depicted in Fig. 1.

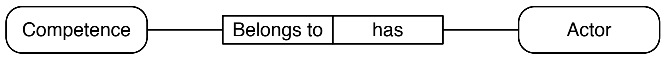

**Fig. 1.** Example of a lexon in some context $\gamma$.

An ontological commitment to such a "lexon base" means selecting or reusing a meaningful set of facts from it that approximate the intended conceptualization, followed by the addition of a set of constraints, or rules, to this subset. This commitment process is inspired by the fact-based database modeling method NIAM/ORM [11].

DOGMA was extended with a "Meaning Evolution Support System", resulting in an interorganizational ontology engineering methodology called DOGMA-MESS [3]. The common ontology is produced in a number of iterations, where each iteration consists of four stages [12]:

1. *Community Grounding*: The core domain expert and the knowledge engineer identify the relevant key concepts with their relations and externalize them in an ontology. This ontology contains conceptualizations common to and accepted by the community.
2. *Perspective Rendering*: In this phase, each stakeholder's domain expert renders its perspective on the common ontology, by specializing it to their idea.
3. *Perspective Alignment*: In perspective alignment a new proposal for the next version of the common ontology is produced. Relevant material from both the common ontology and the different perspectives is gathered and aligned. The methodology allows domain experts to render their view not only by specialization, which allows new definitions to be created, but also by generalization. A consequence of this "creative chaos" is that the alignment process is far from trivial. During that process all the domain and core domain experts need to collaborate.
4. *Perspective Commitment*: In this phase the part of the ontology that is aligned by the community forms the next version of the common ontology. All participants finally commit their instance bases (e.g. databases) and applications to the new version of the ontology.

A *template* is provided to the different communities in order to model the relevant parts. This template can be described as a generalization of a pattern, either found in the models or through experience of the knowledge engineers. An example of such a template is given in Fig. 2. Given a template and a taxonomy, domain experts can *specialize* the relations in the template's by replacing the concepts in that relation by one of its children in the taxonomy. Domain experts can also extend the taxonomy by introducing new concepts. The latter is also called a *specialization* of a template. A revision of the template is needed when domain experts introduce new relations rather then specialize the given set of relations, thus making templates also subject to change and evolution.

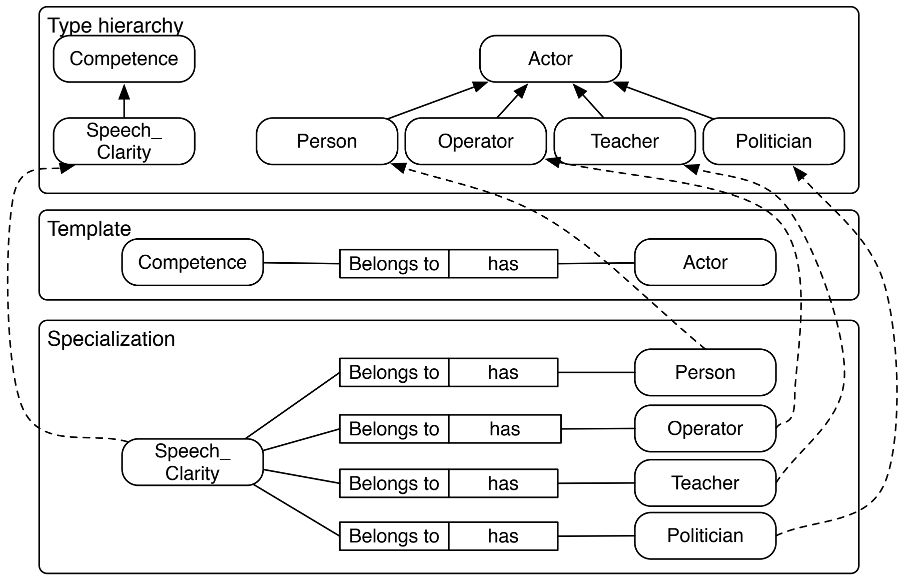

**Fig. 2.** Example of how an organization or community can specialize a given template using concepts from a taxonomy.

### 3.2 Algorithms and Experiment

In this section we will present the algorithms. The algorithms are illustrated with a running example, based on an experiment held at STARLab. Their combined output gives means to answer the three questions formulated in Section 2. Note that there is no one-to-one mapping between algorithms and questions.

**Experimental Setup** The experiment involved five groups of students representing fictive organizations with different core businesses: bug tracking, versioning, document annotation, source code annotation and module management.

The goal was to model a part of the open source community. The first group provided the basis of a first version of a template, which is depicted in Fig. 3.

The template was given to the remaining four organizations, which had to specialize it. These specializations had to be examined for differences, similarities and conflicts.

Files and images concerning the template, specializations and descriptions of each of the organization's core businesses as well as the output of the algorithms can be found at `http://wilma.vub.ac.be/~chrdebru/combek/index.html`

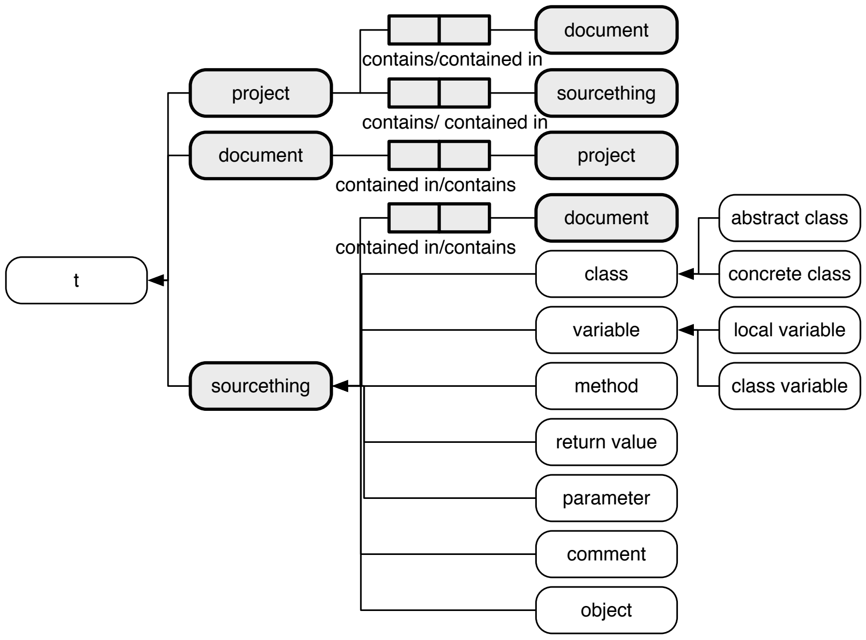

**Fig. 3.** First version of the template based on the first organization's model. The arrows denote the type hierarchy. The terms and role which are colored denote the binary relations that make up the template.

**Algorithm 1** This algorithm looks at the different specializations to determine which relations of the template were specialized or generalized. We first define a function $relationsOf : C \times T \to L$, where $C$ is the set of concepts, $T$ is the set of templates and $L$ is the set of all possible sets of lexons. The function $relationsOf$ returns a subset of lexons in a template $t \in T$ where a concept $c \in C$ either plays the role or the co-role.

Given a template $t \in T$ and a set of specializations $S = \{s_1, \ldots, s_n\}$ by the different communities, the algorithm works as follows:

- For each concept $c \in C$ in $t$ playing either the role or co-role in a binary relation that needs to be specialized:
  - $rels \leftarrow relationsOf(c, t)$
  - For each $s_i$, where $i = 1 \rightarrow |S|$
    - if $c \in s_i$ then
      - Look at the taxonomic children of $c$ in $s_i$, either from the common ontology or introduced by the organization. If those children were used in a specialization of one of the relations in $rels$, we have found a specialization.
      - Look at the taxonomic parents of $c$ in $s_i$. If the concept $c$ in $s_i$ has relations not appearing in $rels$, but appearing in one of its taxonomic parents[2] and that same relation does not appear in $T$, we have found a generalization.

This algorithm, which returns the specializations and generalizations by the organizations, is straightforward and answers the first of the three questions.

**Results of Algorithm 1** Only one organization enriched the taxonomy under one of the existing terms (see Fig. 4). However, none of the organizations had actually specialized any of the provided binary relations. This was mainly due to the nature of the experiment, where there was no clear 'common goal' for each of the stakeholders. The organizations were asked to model a part of the open source community instead, which they did by introducing new concepts and relations needed for their own core business.

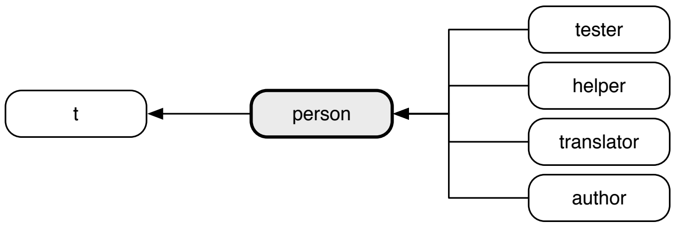

**Fig. 4.** Some of the terms introduced under the concept 'person' by one of the organizations.

**Algorithm 2** The second algorithm will look for candidate concepts in the different views, which might get introduced in a new version of the common ontology. The second algorithm also provides an answer to the question of which are the stable parts in the ontology.

[2] The taxonomic parents in the specializations can differ because organizations are allowed to change the concept's place in the type hierarchy.

For this process we define a function $poc : C \times O \to C$, where $C$ is the set of all concepts, $O$ the set of all ontologies, templates and specializations. The function searches a concept in an ontology $o \in O$ with a certain label and returns its parent in the type hierarchy, *poc* stands thus for *Path of Concept.* If the ontology does not contain such a concept, then the functions returns nothing.

We construct a matrix where we create a row for each concept $c \in C$ and a column for the template and each specialization. Then in each cell of the matrix we output the result of the function *poc* for that particular concept and template/specialization.

| Concepts | Template $T$ | Specilization $S_1$ | ... | Specialization $S_n$ |
|---|---|---|---|---|
| $label_1$ | $poc(label_1, T)$ | $poc(label_1, S_1)$ | ... | $poc(label_1, S_2)$ |
| $label_2$ | $poc(label_2, T)$ | $poc(label_2, S_1)$ | ... | $poc(label_2, S_2)$ |
| $label_3$ | $poc(label_3, T)$ | $poc(label_3, S_1)$ | ... | $poc(label_3, S_2)$ |
| ... | ... | ... | ... | ... |

We can determine what an organization did for a certain concept: introduction, removal (if the concept does not appear in the column of the organization, but does appear in the column of the template) or nothing at all.

We can now take possible actions for the concepts given a certain threshold $\epsilon$. We assign each concept a score, which is the number of times the concept appears in the template or one of the specializations divided by the number of specializations + 1 (for the template). If the majority of the organizations removed a certain concept, the concept's score will be lower than $\epsilon$ and hence become a candidate for dropping. If a certain number of organizations introduced a concept (not considering the taxonomy), the score goes up. When the score of a concept is greater than or equal to $\epsilon$, the concept becomes a candidate for introducing.

As already mentioned, this algorithm answers two of the three questions, namely which concepts are considered interesting for a next iteration and which are the stable parts in the specializations. The latter is answered by looking at the results of the *poc* function. If most (if not all) concepts have the same parent, we consider them stable.

**Results of Algorithm 2** The second algorithm showed that, given a certain threshold, both *version* and *person* could be added to a next version of the ontology. Both terms are also candidates for future iterations of the process. The resulting matrix (see Table 1 for a part of the matrix) of the second algorithm also showed that most concepts in the template are stable. One organization restructured the template's taxonomy; making only everything below the concept *sourcething* truly stable. Note that concepts such as *account* and *admin*, which do not occur in the template found in Fig. 3, have been introduced by one or more organizations.

| | Template | $\text{Org}_1$ | $\text{Org}_2$ | $\text{Org}_3$ | $\text{Org}_4$ | Score |
|---|---|---|---|---|---|---|
| _Object | | | | | T | 0.2 |
| Abstract class | Class | Class | Class | Class | Class | 1.0 |
| Account | | | Person | | | 0.2 |
| Admin | | | Person | | | 0.2 |
| Author | | | | | Person | 0.2 |

**Table 1.** Part of the matrix constructed by the second algorithm.

When interviewing the different groups about the taxonomy; the groups that did not alter the type hierarchy argued that such operations would have cost too much if commitments to the ontology were already made. The group that changed the type hierarchy found that such (possibly costly) operations in early stages of ontology engineering are ground, provided if they would benefit the reusability and correct level of abstraction of the ontology in the long run.

**Algorithm 3** The third and last algorithm will look for candidate relations between concepts, which will be introduced in a next version of the common ontology. This algorithm uses the output of the second algorithm.

The algorithm works as follows: given a second threshold $\alpha \leq \epsilon$, for each concept $c$ marked as a candidate for introduction in the previous algorithm:

– List all the relations of that concept for each specialization.
– Register the occurrences of each of these relations across all specializations, solely looking at the lexical representation of these relations (or lexons).
– If the average occurrence of a relation is greater than or equal to $\alpha$, the relation and (if not yet introduced) the other concept, are included in a next version of the template as well.

**Results of Algorithm 3** Two organizations introduced the term *version* and *person* was introduced by three organizations. Due to the very different natures of the core businesses; the organizations have not introduced similar relations between concepts.

### 3.3 Meaning Negotiation

The results of these three algorithms were taken to the meaning negotiation session, where they were discussed. During this discussion some problems arose, which became points of the agenda of that meeting. One of these problems was the difference between the terms *version* and *revision*, since the results marked the term *version* as a candidate for introduction and not *revision*. Some organizations argued they denote the same concept. After discussing the context of both terms (and the relations around the concepts), all organizations agreed they were different.

One organization restructured the type hierarchy to benefit future reuse, while others considered this too much of a risk (and a cost) when commitments to that ontology were already made. Since the majority of the stakeholders did not want to risk that cost, the changes to the type hierarchy were ignored. We also found that providing a very basic taxonomy would have encouraged the organizations to introduce new terms more neatly in a type hierarchy.

## 4 Discussion

One of the most difficult tasks in ontology engineering is to find a consensus between different stakeholders while trying to define a common ontology. Meaning negotiation, where problems and conflicts within the different views among the stakeholders are discussed, is therefore an important process. This process can be guided by answering three questions about the different views: (1) what relations have been generalized or specialized, (2) what are the stable parts in the ontology and (3) what are new interesting concepts that have to be modelled in future iterations. These three questions will point out the problems, which can be used to construct an agenda for the meaning negotiation process.

In this paper we extended a state of the art collaborative ontology engineering methodology, called DOGMA-MESS, with three algorithms answering these proposed questions. These algorithms were applied in an experiment where groups of students simulated the methodology to model a part of the open source community by representing fictive organizations. The list of similarities and conflicts between the different models was limited due to the very different core business of the organizations, but the algorithms showed their potential in constructing an agenda for a meaning negotiation session.

## 5 Future Work

The algorithms were currently applied on the models at lexical level, applying them while also taking into account the type hierarchy or glossaries would add semantics to these algorithms.

Even though the results proved promising, the constructed algorithms still need to be validated with a use case of substantial size. Such an experimental setup will also result in an implemenation of the algorithms.

Another trail worth investigating is applying the extention to other collaborative ontology engineering methodologies like HCOME [5] and DELIGENT [8]. The difference between those methodologies and DOGMA-MESS is the explicit use of templates, which is used as input by the algorithms, by the latter.

**Acknowledgements**

We would like to thank our Professor Robert Meersman and everyone at VUB STARLab for aiding us with this experiment as well as proposing us to try and publish the results. We would also like to thank Simon Janssens, a Computer Science student at the VUB, for assisting us with this experiment.

## References

1. Gruber, T.R.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition 5 (1993) 199-220
2. Jarrar, M., Meersman, R.: 3. In: Ontology Engineering - The DOGMA Approach. Volume 1 of Advances inWeb Semantic, A state-of-the Art SemanticWeb Advances in Web Semantics IFIP2.12. Springer-sbm (2007)
3. de Moor, A., De Leenheer, P., Meersman, R.: DOGMA-MESS: A meaning evolution support system for interorganizational ontology engineering. In Schärfe, H., Hitzler, P., Ohrstrom, P., eds.: Conceptual Structures: Inspiration and Application. Volume 4068., Aalborg, Denmark, Springer Berlin / Heidelberg (2006) 189-202
4. Kalfoglou, Y., Schorlemmer, W.M.: Ontology mapping: The state of the art. In Kalfoglou, Y., Schorlemmer, W.M., Sheth, A.P., Staab, S., Uschold, M., eds.: Semantic Interoperability and Integration. Volume 04391 of Dagstuhl Seminar Proceedings., Dagstuhl, Germany, Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI) (2005)
5. Kotis, K., Vouros, A.: Human-centered ontology engineering: The hcome methodology. Knowledge and Information Systems 10 (2006) 109-31
6. Kunz, W., Rittel, H.: Issues as elements of information systems. Working Paper 131, Institute of Urban and Regional Development, University of California, Berkeley, California (1970)
7. Pinto, H.S., Staab, S., Tempich, C.: Diligent: Towards a fine-grained methodology for distributed, loosely-controlled and evolving engingeering of ontologies. In: In Proceedings of the 16th European Conference on Artificial Intelligence (ECAI, IOS Press (2004) 393-397
8. Meersman, R.: Semantic ontology tools in is design. In Ras, Z.W., Skowron, A., eds.: ISMIS '99. Volume 1609 of Lecture Notes in Computer Science., Springer (1999) 30-45
9. Meersman, R.: Reusing certain database design principles, methods and design techniques for ontology theory, construction and methodology. Technical report, VUB STARLab (2001)
10. Halpin, T.A.: Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2001)
11. De Leenheer, P.: Towards an ontological foundation for evolving agent communities (2008) Invited paper for IEEE ESAS 2008.