Ontology mapping specification in description logics for cooperative systems

Thibault Poulain, Nadine Cullot, Kokou Yétongnon

Ontology mapping specification in description logics for cooperative systems

T. Poulain , N. Cullot, K. Yétongnon

Laboratoire d'Electronique, Informatique et Image,
Université de Bourgogne, Bat. Mirande - BP 47870 21078 Dijon Cedex France

RÉSUMÉ. *Le développement rapide du Web sémantique est lié à la spécification de plus en plus d'ontologies. Celles-ci permettent de modéliser des connaissances agréées par des communautés de personnes concernant des domaines ou des tâches spécifiques. Le même domaine décrit par deux communautés distinctes sera modélisé de façon différente. Les systèmes coopératifs visent à rendre les informations provenant de différentes sources disponibles au-delà de leurs divergences. Pour cela, ils doivent aligner, fusionner ou intégrer ces ontologies. La découverte de mappings est un point clé dans la résolution efficace des hétérogénéités entre ontologies. Nous développons une architecture qui connecte des systèmes d'information via des ontologies, avec comme objectif la résolution de requêtes complexes. Le but de notre article est de décrire les grandes lignes de cette architecture, après avoir présenté la méthodologie utilisée pour mettre en correspondance les diverses ontologies rencontrées.*

ABSTRACT. *The rapid development of the semantic Web is associated with the specification of various ontologies which formally represent agreements of communities of people on specific domains or tasks. The same knowledge formalized by different people leads to heterogeneous representations. Cooperative systems, which aim to make knowledge from various sources available in spite of their heterogeneities, need to align, merge or integrate the ontologies used to model the information sources. Furthermore, the resolution of differences among ontologies is necessary to process queries or use web services in distributed heterogeneous environments. Mapping discovery is a key issue to allow efficient resolution of heterogeneity. We develop an architecture for mapping different systems associated with ontologies. In this paper we present the key components and the underlying concepts of a framework and approach for comparing and matching different ontologies.*

MOTS-CLÉ : *Mappings, ontologies, web sémantique, calcul de similarité estimation*

KEYWORDS: *Mappings, ontology, semantic web, similarity estimation.*

1. Introduction

The definition of the "Semantic Web" introduced by Berners-Lee et al. [2] in 2001 finds its origin in the rapid development of Web technologies and the difficulty to manage the ever increasing amount of web based data sources.. The development of models, methodologies, tools and architectures for automated access to data is a major challenge to turn the Web into a tool to efficiently support information sharing among not only people but also computers.

Ontologies are an essentially technology for the semantic Web. They allow the specification of a domain's semantic. Their various models of representation are based on approaches from logics or from databases [3]. Amongst logics based approaches, OWL-DL, a description logics language specified by the W3 Consortium is one of the emerging standards. Description logics are a family of languages that allow formal representation of knowledge and provide reasoning tools for classification and consistency checking.

The rapid development of multiple ontologies on various domains creates the need for methodologies to map or interconnect them into cooperative systems. Cooperative systems must address data heterogeneity issues which arise from syntactic, structural or semantic differences in data sources. The management and resolution of semantic heterogeneity is a significant research challenge which requires methodologies and tools to specify semantic associations. Two main approaches can be used to achieve interoperability between heterogeneous systems. One approach consists in merging two or more ontologies to create a single ontology to which all the data are associated. Another approach consists in using mappings to establish semantic correspondences among the source ontologies. The latter approach preserves the local knowledge. Queries submitted to a specific system can be translated by mappings and executed on other systems. Mappings generation is a critical element for cooperative systems. It can be automatic, semi-automatic or manual. In all cases, this process is used to resolve

the various heterogeneities existing between the ontologies.

Heterogeneities between knowledge bases has been extensively studied in the database domain. They can be divided in different categories, namely syntactic structural and semantic heterogeneities. Syntactic heterogeneity arises from the use of different representation models. Structural heterogeneity is due to differences of conceptual choices. Semantic heterogeneity arises from differences in the intended meanings associated to the terms used to describe information.

The aim of the paper is to propose a framework for querying cooperative systems. The data in the local sources are associated with a local ontology that can be mapped to one or more reference ontologies corresponding to different application domains. The focus of the paper is on the discovery and the representation of mappings from the local ontologies to a reference ontology. The ontologies and mappings are described in Description Logics. A running example is given to illustrate the mapping discovery process.

The paper is organized as follows. Section 2 gives a brief reminder of description logics and some related works on mappings. Section 3 presents the ontology mapping approach. Section 4 describes the proposed cooperative framework. Finally, section 5 concludes the paper.

2. Related Works

In this section, we present a brief overview of description logics and discuss existing mapping tools

2.1. Description logics

Description Logics are a family of terminological formalisms to specify and reason on knowledge. Their expressiveness depends on the considered constructors. They rely on two basic notions: concepts and roles. Complex concepts and roles can be built from the basic concepts using the DL constructors. A detailed presentation of DL can be found in [1].

The basic $\mathcal{ALC}$ DL provides the following concept constructors: $\neg C$ (negation), $C \cap D$ (conjunction), $\forall R.C$ (value restriction) and $\exists R.C$ (existential restriction) where C and D are concepts and R is a role.

The $\mathcal{SHIQ}$ DL [10] extends the basic $\mathcal{ALC}$ DL to provide an expressive language. $\mathcal{SHIQ}$ DL [10] is implemented in the RACER prover[8]. The elementary descriptions are the atomic concepts and roles, the universal concept T and the bottom concept $\perp$. In addition, the $\mathcal{SHIQ}$ DL includes the inverse role $\mathcal{I}$, role hierarchies $\mathcal{H}$ and qualifying number restrictions $\mathcal{Q} \geq n$ R.C and $\leq n$ R.C. Complex concepts can be defined using the inclusion ($\subseteq$) and equivalence ($\equiv$) axioms. For example, the following formula states that PhD students are included in the set of teachers who work in a laboratory.

$$\text{PhDStudent} \subseteq \text{Teacher} \cap \exists \text{works in.Lab}$$

2.2. Mapping Tools

Significant works have been done on mapping specifications. We focus below on four relevant projects: GLUE, MAFRA, the Prompt suite project and QOM. They offer tools for the discovery and representation of mappings between ontologies.

The **GLUE**[5] project proposes an automatic mapping creation system which uses instances to determine which concepts should be matched. It relies heavily on learners and training methods borrowed from artificial intelligence to achieve concepts matching. Learners are trained with the instances of one ontology to be able to classify input data as belonging or not to a specific concept. The training phase used to compute mappings makes the addition of new ontologies on the fly difficult. Furthermore, the fact that the method works on instances reduces the number of cases where it can be applied. For example, It cannot be used to map ontologies with no instances.

The **MAFRA**[16] (MApping FRAmework) project provides an automatic mapping creation tool as a plug-in to the KAON (KArlsruhe ONtology) project. MAFRA defines the notion of semantic bridges. It describes relations between ontology entities such as concepts, relations and attributes in a detailed way. The strength of this work is the specification of mappings using these semantic bridges allowing the representation of complex mappings. However, there is no automatic method for to discovering and writing mappings using the semantic bridge formalism.

The **Prompt tools suite**[13] consists of several components which are integrated to Protégé [14] as plug-ins. The project proposes a tool allowing the merging of ontologies. It is able to check the mappings specified by users to detect possible conflicts and it uses this input to suggest new mappings. Prompt is a semi-automatic tool where human interaction is needed at each step. The mapping propagation rules rely only on the taxonomic hierarchical structure of the ontologies. A problem with this approach is that ontologies are not always organized into a single taxonomic tree.

QOM[6] (Quick Ontology Mapping) is a semi-automated system which aims at discovering mappings in a rapid way, rather than maximizing the accuracy of those mappings. QOM is based on a mapper called NOM (Naive Ontology Mapper) and uses various similarity estimation methods as well as combination of those methods, as in so-called hybrid mapping systems.

Taking into account the specificities of each studied systems, we retain some relevant requirements for our architecture.

- *Evolutivity* which is needed to allow new methods of mapping estimation to be added. Our architecture supports the definition and addition of different similarity estimation methods to customize the mapping discovery phase to optimize results depending on the mapped ontology.
- *Automation* of the mapping process which reduces human intervention and decreases the time required to map any new ontology to the cooperative system. New systems should be able to connect to the central architecture with a waiting time as small as possible.
- *Instances* should not be used to support mapping discovery, since methods based on instances require a somewhat greater time to be completed. A more suitable approach is to map the different ontologies to a generic reference ontology without instances.

3. Ontology mapping for a cooperative system

3.1. Overview of the approach

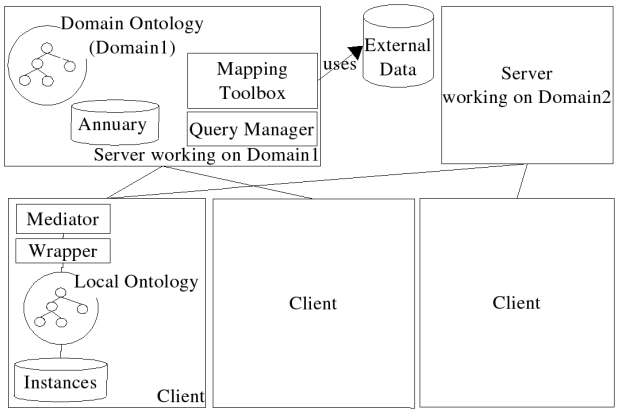
Figure 1 presents the mapping architecture, consisting of a network of client and server systems. Each server describes knowledge about a specific domain. Clients which want to share their data map their ontology to the domain ontologies of the relevant servers. When a client has to answer a query, it propagates it to the servers to which it is connected. Those servers then query the other clients and send back the answer set to the first client.

Each server specifies the mapping rules which any client must follow to connect. These rules are gathered in the "Mapping Toolbox". It provides specific mapping information to the clients wishing to connect to the server, as well as references to semantic taxonomies, like WordNet [7] or a dictionary, which are the external data used to perform the mappings. The domain ontology held by a server has no instances.

Clients are systems which store local ontologies and the associated data. They interact with servers and indirectly with any other client connected to the same server. To join the cooperative system and share its contents, ,a client needs to map its ontology to the ontology of one or more servers.

We do not study here what types of data are associated to local ontologies. It could be one or several databases or XML documents. Since we only interact with the data through a local ontology, we will not deal further with the way data are stored.

Figure 1. Overall Architecture



3.2. Mapping creation

To illustrate the process of mapping the ontologies of a server and a client, we consider a running example that models information sharing in a university domain. We define two ontologies called O1 and O2 and described in Figure 2 and 3. O1 describes the knowledge of the server about the university structure while O2 refers to the client's ontology.

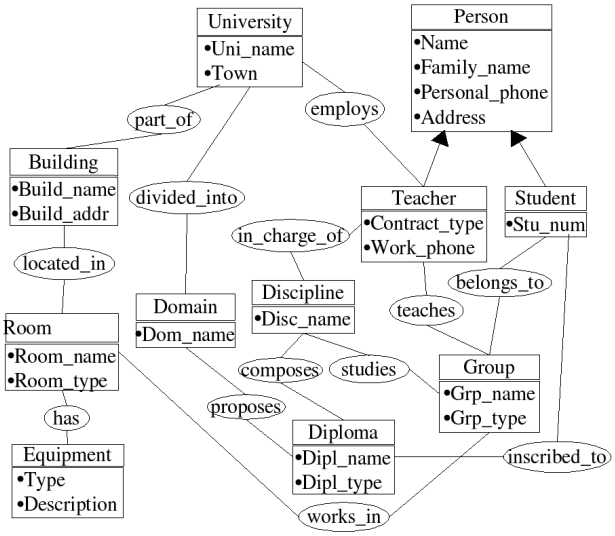


Figure 2: Domain Ontology on the Server (O1)

Both graphical representations show concepts as a box which displays the name and the properties of the concept. Roles are represented by a circle. We assume that both the client and server use DL to describe their ontologies. Figure 4 shows a partial specification of the two ontologies O1 and O2 in DL.

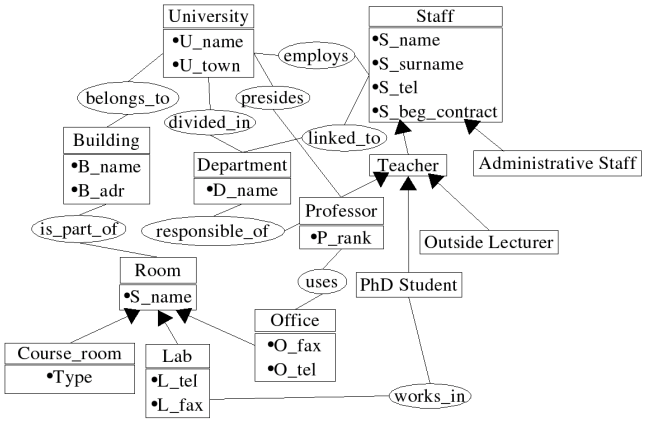


Figure 3: Local Ontology on the Client (O2)

The mapping creation methodology consists of two phases: watching and refining. Details about the mapping process depend on the rules, which are specific to each server. These rules describe the mapping methods and point to any needed external data. Each server starts with a basic set of rules, which can be extended by experts. They can specify the similarity estimation methods recommanded by the server, define the way their results are combined and define new ways of propagating similarities during the refining step.

O1.Person ≡	∃ O1.Name.O1.PersonName
	∩ ∃ O1.Family_name.O1.PersonFamilyName
	∩ ∃ O1.Personal_phone.O1.PersonTelNumber
	∩ ∃ O1.Address.O1.PersonAddress
O1.Teacher ⊆	O1.Person
	∩ ∃ O1.Contract_type.O1.TeacherContract
	∩ ∃ O1.Work_phone.O1.TeacherTelNumber
	∩ ∃ O1.teaches.O1.Group
	∩ ∃ O1.in_charge_of.O1.Discipline
O1.Student ⊆	O1.Person
	∩ ∃ O1.Stu-num.O1.StudentNumber
	∩ ∃ O1.belongs_to.O1.Group
	∩ ∃ O1.inscribed_to.O1.Diploma
O2.Staff ≡	∃ O2.S_name.O2.StaffName
	∩ ∃ O2.S_surname.O2.StaffSurname
	∩ ∃ O2.S_tel.O2.StaffTelNumber
	∩ ∃ O2.S_beg_contract.O2.StaffContractNumber
O2.Professor ⊆	O2.Teacher
	∩ ∃ O2.P_rank.O2.ProfessorGrade
	∩ ∃ O2.responsible_of.O2.Department
	∩ ∃ O2.uses.O2.Office
O2.OutsideLecturer ⊆	O2.Teacher
O2.PhDStudent ⊆	O2.Teacher
	∩ ∃ O2.works_in.O2.Lab
O2.Office ⊆	O2.Room
	∩ ∃ O2.O_fax.O2.OFaxNumber
	∩ ∃ O2.O_tel.O2.OTelNumber
O2.Lab ⊆	O2.Room
	∩ ∃ O2.L_fax.O2.LFaxNumber
	∩ ∃ O2.L_tel.O2.LTelNumber

Figure 4: Partial description of the ontology

3.2.1. Matching

During this phase, a similarity value is assigned to each pair of concepts from each ontologies being matched. The estimated similarity can be used to locate semantically close concepts, and therefore, mappings. Pair of concepts with a high similarity value can be considered as equivalent. One or more similarity computation methods which we list below can be used

- *Simple comparison* is the simplest method. It considers two strings of characters as similar only if they are exactly the same.
- *Synonyms* uses a list of equivalence to determine similar words, by example by using the WordNet links of synonymy.
- *Affix* checks if a string starts or ends with another one, by example Phone for Telephone.
- *Edition distance* method considers the number of modifications needed to transform one string into the other one.
- *Phonetical* method compares the phonetic similarity between two strings.(Do and Rahn[4])
- *N-Gram* computes the number of the same n-gram instances between two strings. N-grams are a string of n letters.

- *Graph Distance* calculates the distance between the words inside a reference graph.(Jiang and Conrath[11])
- *Definition* computes the similarity of two strings based upon the similarity of each word's definition. It requires a reference dictionary listing the definitions of all terms met.(Lesk[12])
- *Expansion* determines similarities between acronyms and their expanded meaning.
- *Translation* is a variant of the *Synonyms* method which is used to estimate similarities between concept names described in different languages.
- *Hybrid* methods combine two or more of the methods cited above and return a global result.

All those methods can be proposed by any given server, and must be eventually combined to obtain a single similarity estimation value.

3.2.2. Refining

The refining phase is used to improve the consistency of the results obtained in the previous step. It is an iterative process which takes as input the results from the matching phase. The similarity values are modified using neighbourhood rules, which are used to determine if the similarity value between two concepts must be increased or decreased based on neighbourhood similarities. These rules can be either generic or knowledge-domain specific. For example, if the concepts C1and C2 from Ontology 1 and Ontology 2 respectively match according to the computed similarity then it is likely that their respective father concepts in the hierarchy match also, possibly allowing us to infer their similarities. If the modification makes the fathers' similarity value high enough, the same rule can in turn be applied to the upper step of the hierarchy for the next iteration.

The rules are not necessarily linked to taxonomic relations between concepts. The neighbours of the concepts can affect the similarity results independently of the role they are linked to. Figure 5 shows an example of a non-taxonomy rule. Consider concepts Person from the reference ontology and Staff from the local ontology, their respective neighbourhoods include all concepts directly linked to them by any role. The proportion of matching terms of the two groups is compared to a threshold. If the matching is high enough, then it is likely that the concepts match. Their similarity is increased as a consequence.

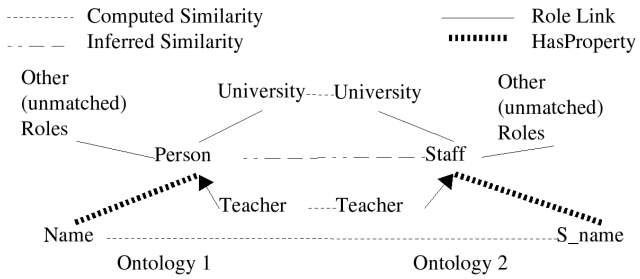


Figure 5: Neighbourhood affects concept similarity

In our system, experts are allowed to design knowledge-domain rules which represent logical statements about concept organization. They are linked to specific notions, as Teacher or University and we associate them with the concepts of our reference ontology. For example, a specific rule of the university domain could be: If the computed similarity suggests that Concept2 matches Teacher, then the probability that his father Concept1 matches Professor which is a child concept of Teacher decreases.

Each iteration takes the neighbours from each concept and modifies the similarities as required. The process is then repeated as long as needed. This process does not always reach a stable state. To avoid this problem and to reduce the execution time, the maximal number of iterations can be set to a fixed value as explained by Hameed Preece and Sleeman[9]. The following mapping is obtained through the refining step. As the concepts named Teacher from both ontologies match, the similarity between the fathers of concepts Person and Staff

has been increased, enough to match both concepts. Experts finally transform the equivalence into a subsumption: $O2.Staff \subseteq O1.Person$

After several iterations, the concepts Staff and Person are matched. There is one rule stating that if we consider a concept whose father matches with another concept, then the considered concept is subsumed by the matched one. Thus all children of the concept Staff are subsumed by the concept Person.

$O2.AdministrativeStaff \subseteq O1.Person.$

At the end of the automatic phases, DL provers or other tools can be used to check the consistency of the produced mappings. Extensions can also be done manually by an expert who can modify the produced mappings or create complex mappings that cannot be discovered automatically.. The equivalence defined between the two concepts Student and PhDStudent $O1.Student \equiv O2.PhDStudent$ is converted into the subsumption $O2.PhDStudent \subseteq O1.Student$

4. System description

This section presents the functional and architectural details of the server and client components introduced in the previous section. Figure 1 shows a global view of the architecture.

4.1. Client System

As shown in Figure 6, a client is divided into three parts: Data Layer, Wrapper and Mediator.

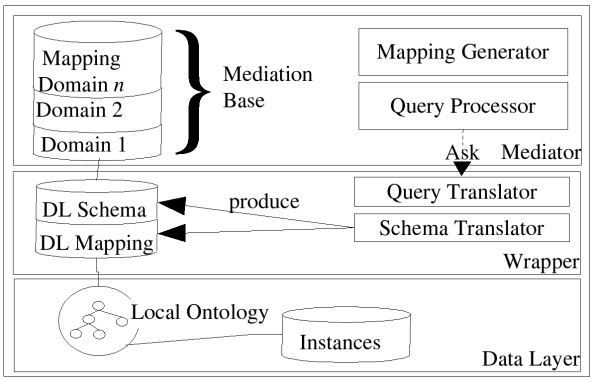


Figure 6 : Details of the client

The **Data Layer** is the core of the client system. It is composed of an ontology and its instances. The language of the ontology is not defined, as we assume that we can translate it to Description Logics with the wrapper.

The **Wrapper** is a translation interface between the language of the local ontology and DL. The wrapper generates a DL representation of the local ontology, with agreements between the original ontology and its translation. The representation (called DL Schema) and the correspondences (called DL mapping) are then stored to avoid the translation each time a user wants to solve a query, as well as allowing the mappings enhancement by experts.

Queries submitted to the client system (by opposition with queries asked on the server) must be translated by the wrapper by its Query Translator. They are then transferred to the query processor, one component of the mediator.

The **Mediator** is composed of three parts. The mapping generator uses the DL Schema, the reference ontology from the chosen server as well as its similarity computation variables to generate the mapping from the local ontology towards a given server. Resulting mappings are stored inside the mediator. Several mappings corresponding to several server systems are stored in the mediation base, which is the second part of the mediator.

The **query processor** combines the query answers received from the server system and the answers from the client system.

4.2. Server System

As shown in Figure 7, the server system is composed of a Reference Ontology, a Mapping Toolbox, a Query Manager and an Annuary.

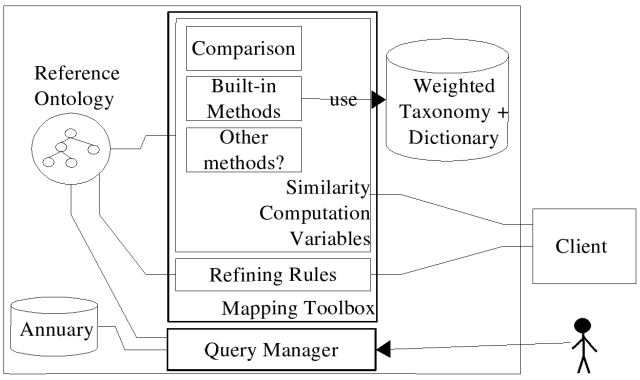


Figure 7: Details of the server

The ontologies of the client systems are mapped to the server Reference Ontology. The reference ontology describes common information over a specified knowledge domain. We insist on the fact that the goal is not to match all data from ontologies of the clients but only the ones that are relevant domain-wise. However, nothing prevents the clients to be mapped to several server systems.

The **mapping toolbox** describes the mappings rules: methods, external data and refining rules.

Methods are used as plug-ins to compute similarity between two terms. A value is associated with each method to define its weight in the whole estimation process [16].

External data are reference items used to compute similarity. They are usually one reference taxonomy and a reference dictionary.

Refining rules are a list of conditions to be checked in while processing the refining phase. Experts can extend the basic set of rules provided with specific methods to estimate similarity, with the way to combine these estimated similarities and with semantical rules on how to adjust those results according to the concepts' neighbourhood.

The **Query Manager** centralizes queries and dispatches them to clients which are likely to provide relevant answers. The query manager uses the annuary to know which clients it should ask.

Queries can be submitted by users to a client or directly to a server. If the query is sent to a client, the query processor solves the part related to its local ontology, then it sends the results and the query to the query manager of the server. The query is translated in OWL-QL [15] and adapted for the reference ontology with the help of the domain mappings stored in the client.

The server's query manager divides the query into smaller and unnested queries. The resulting subqueries are then sent to any client likely to own relevant information. Relevant clients are determined using the annuary content. For example, consider a query for finding all the laboratories managed by some universities. If one client's local ontology does not take into account the research activity of the university, it is irrelevant to query this client. The query manager then only asks the relevant clients. Once a query is solved by a client, the results are sent back to the server which must organize and complete them.

The **Annuary** contains a list of concordances between the generic ontology and each local ontology. Concordances are not a complete description of the mappings but a list of which client ontologies are linked to each concept of the reference ontology. Thus, we are able to query only clients which can bring information.

5. Conclusion

After a quick overview on the ontologies domain, we presented some existing mapping tools with their strength and weakness.We proposed a mapping method designed to be integrated in an ontology management structure that we also presented. Our proposed architecture uses several features. It is a scalable structure, able to cope with new methods of automatic mappings.

Restrictions on the domain of the server's ontology allow the clients to know what kind of information can be found on a given server. Moreover, it prevents the ontology on the server to become too big with new information constantly added, even if the server supports ontology improvement.

The queries are managed cooperatively by the server and the clients, notably through the use of an annuary.

6. Références

[1] Baader F., Calvanese D., McGuinness D., Nardi D., and Patel-Schneider P., editors. The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, 2003.

[2] Berners-Lee T., Hendler J., and Lassila O. The Semantic Web. Scientific American, May 2001.

[3] Cullot N., Parent C., Spaccapietra S., and Vangenot C. Ontologies : A contribution to the DL/DB debate. In Proceedings of the 1st International Workshop on Semantic Web and Database (SWDB'2003) Co-located VLBD'2003, pages 109130, 2003.

[4] Do H. and Rahm E. COMA - A System for Flexible Combination of Schema Matching Approaches. In 28th International Conference on Very Large Data Bases, Aug. 2002.

[5] Doan A., Madhavan J., Domingos P., and Halevy A. Ontology matching: A machine learning approach. In Staab S. and Studer R. [17], pages 385404.

[6] Ehrig M. and Staab S. Efficiency of Ontology Mapping Approaches. In International Workshop on Semantic Intelligent Middleware for the Web and the Grid at ECAI 04, Valencia, Spain, Aug. 2004.

[7] Fellbaum C., editor. WordNet: An Electronic Lexical Database. MIT Press, May 1998.

[8] Haarslev V. and Müller R. Racer system description. In Proc. of International Joint Conference on Automated Reasoning, IJCAR o 2001, pages 701705. Springer-Verlag, 2001.

[9] Hameed A., Preece A., and Sleeman D. Ontology reconciliation. In Staab S. and Studer R. [17], pages 231250.

[10] Horrocks I., Sattler U., and Tobies S. Practical reasoning for very expressive description logics. Logic Journal of the IGPL, 2000.

[11] Jiang J. and Conrath D. Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In International Conference Research on Computational Linguistics (ROCLING X), Aug. 1997.

[12] Lesk M. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In SIGDOC '86: Proceedings of the 5th annual international conference on Systems documentation, pages 2426, New York, NY, USA, 1986. ACM Press.

[13] Noy N. Tools for mapping and merging ontologies. In Staab S. and Studer R. [17], pages 365384.

[14] Noy N., Fergerson R., and Musen M. The knowledge model of Protege-2000: Combining interoperability and flexibility. In EKAW, 2000.

[15] Fikes R., Hayes P., and Horrocks I. OWL-QL – A Language for Deductive Query Answering on the Semantic Web. 2003

[16] Silva N. and Rocha J. Service-Oriented Ontology Mapping System. In Proceedings of the Workshop on Semantic Integration of the International Semantic Web Conference, Sanibel Island (FL), USA, Oct. 2003.

[17] Staab S. and Studer R., editors. Handbook on Ontologies. International Handbooks on Information Systems. Springer, 2004.