FEDERATED MEDIATORS FOR QUERY COMPOSITE ANSWERS

Dong Cheng and Nacer Boudjlida
LORIA-University Henri Poincaré Nancy 1
BP 239, 54506 Vandoeuvre Lès Nancy CEDEX(F)

Keywords: Composite answer, Description Logics, Mediator, complementary.

Abstract: The capture, the structuring and the exploitation of the expertise or the capabilities of an "object" (like a business partner, an employee, a software component, a Web site, etc.) are crucial problems in various applications, like cooperative and distributed applications or e-business and e-commerce applications. The work we describe in this paper concerns the advertising of the capabilities or the know-how of an object. The capabilities are structured and organized in order to be used when searching for objects that satisfy a given objective or that meet a given need. One of the originality of our proposal is in the nature of the answers the intended system can return. Indeed, the answers are not Yes/No answers but they may be cooperative answers in that sense that when no single object meets the search criteria, the system attempts to find out what a set of "complementary" objects do satisfy the whole search criteria, every object in the resulting set satisfying part of the criteria. In this approach, Description Logics (DL) is used as a knowledge representation formalism and classification techniques are used as search mechanisms. The determination of the "complementary objects" is founded on the DL complement concept.

1 INTRODUCTION

In many situations and applications, one is faced to the problem of discovering "entities" that satisfy given requirements. On the other hand, most often, information retrieval in a distributed context, like the World Wide Web, usually lacks selectivity and provides large answer-sets due to the fact that the available information sources are not well structured nor well classified, as attested by the tremendous work about Semantic Web (W3C, 2003a). Moreover, most of the systems usually return yes/no answers (i.e. either they find out answers to a given query or they fail). The work we describe here concerns the publishing of the capabilities of given entities (i.e. what functionalities a given entity is offering, what expertise it has and so on). The capabilities are organized and structured in order to be exploited when searching for entities that satisfy an objective or that meet given needs (searching for Web services that offer given functions, searching for a component in the context of component-based design and component-based programming, searching for a business partner with a given expertise, looking for an employee whose records and expertise satisfy a given work position profile are application examples of our work). It is clear that these needs require the capture and the organization of the entities capabilities together with the classification of the entities and their capabilities. In this work, we adopted an object-oriented knowledge representation using description logics (DL-org, 2003). From the system architecture point of view, we opted for a model based on distributed and cooperative mediators (or traders). A significant originality of our approach resides in the type of answers we aim at providing. Indeed, when no single entity satisfies the search criteria, the systems attempts to determine a set of *complementary* entities who, when grouped together, satisfy the criteria.

The presentation is structured as follows. In section 2, we expose some motivations together with the architecture of the target system viewed as a Knowledge-Based Management System (KBMS). Section 3 briefly introduces elements of description logics and shows how description logics is used for entities capabilities management and retrieval. Concluding remarks are in section 4.

Cheng D. and Boudjlida N. (2004).
FEDERATED MEDIATORS FOR QUERY COMPOSITE ANSWERS.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 170-175
DOI: 10.5220/0002638601700175
Copyright © SciTePress

2 MOTIVATION AND ARCHITECURE

The dynamic discovery of services or capabilities an "entity" offers, has different application domains. Component-based programming, electronic business (*e-business*) and even enterprise knowledge management (Nonaka and Takeuchi, 1995) are among the application domains in which there is a need for the discovery of services or capabilities an "entity" offers. The only syntactic description of an entity's capability (like the *signature* of a software component's service) is not satisfactory when using that description for answering a request: an additional semantic description is required. Moreover, the elicitation of possible relationships among services may contribute to find out "the best" service or the "the best complementary" services that satisfy a search query.

In *e-business*, this approach can be applied in the constitution of business alliances or when looking for business partners. For example, when trying to constitute a business alliance, the system we aim to develop services that may help in retrieving the most appropriate candidate partners. Furthermore, the work depicted hereafter may also serve for semantic-based discovery of Web services (W3C, 2003c) and it is in line with current activities on semantic and ontology for the Web (uddi.org, 2000; Dyck, 2002; W3C, 2003b; W3C, 2003a).

The purpose of this work requires formalizing and structuring the knowledge that concern the "entities" in a given application domain. We opted for Description Logics (DL) (DL-org, 2003; Horrocks, 2002b) as a knowledge representation formalism: it has the notable merit that a single mechanism (the classification mechanism) serves at the same time to build and to query extendible domain descriptions.

As a matter of comparison, in (Han et al., 1999), "entities" are design fragments that are described thanks to keywords, the relationships between the fragments are constituted by metrics that measures the similarity between fragments. In (Borgida and Devanhu, 1999), entities are object-oriented software components and description logics is used, notably, to describe their intended semantics together with possible constraints involving objects methods.

In (Bouchikhi and Boudjlida, 1998), "entities" are software objects and the capabilities of a software object are specified giving a syntactic part (signatures of the methods or operations the object offers) and a semantic part expressed as logical expressions (a precondition and a post-condition are associated with every object's method). The syntactic conformance of a method to a query is trivial and the semantic conformance uses theorem proving techniques.

One should notice that description logics has been used in the database domain (Borgida, 1995; Boudjlida, 1995)[1], not only for querying but also for database schema design and integration (Calvanese et al., 1998), for reasoning about queries (query containment, query refinement, query optimisation, ...) (Beeri et al., 1997). From our concern, description logics is used for query purposes with the special objective to produce more than "Yes or No" results.

From the system architecture point of view, we choose a trader (also called mediator) based architecture (OMG, 1992) very similar to the notion of *discovery agency* in the Web services architecture. In this architecture, an "entity", called *exporter*, publishes (*tells*) its capabilities at one or more mediators sites (see figure 1). Entities, called *importers*, send requests (*queries*) to the mediator asking for exporters fitted with a given set of capabilities.

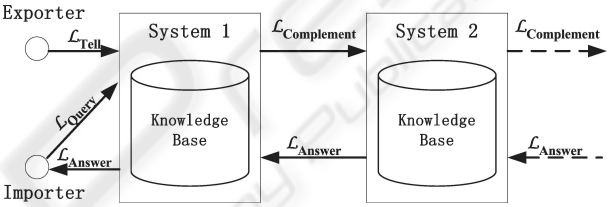

Figure 1: The Mediator-based Architecture

The mediator explores its knowledge base to try to satisfy the query. The capability search process is founded on the exported capabilities and on relationships between them, these relationships being transparently established by the mediator. When some exporters known from the mediator satisfy the query, the references of those exporters are sent back to the importer in $\mathcal{L}_{answer}$. Nevertheless, satisfying the query falls into different cases (Boudjlida, 2002) (see figure 2):

- Case1: There exist exporters that exactly satisfy the query;
- Case2: There exist exporters that fully satisfy the query, but their capabilities are wider than those requested;
- Case3: There exists no single exporters that fully satisfy the query, but when "combining" or composing capabilities from different exporter, one can fully satisfy the query;
- Case4: Neither a single exporter nor multiple exporters satisfy the query, but there exist some exporters that partly satisfy the query;
- Case5: No single exporter nor several exporters fully or partly satisfy the query.

[1] See also (DL-org, 2003) for the proceedings of the various "Knowledge Representation meets DataBase" (KRDB) Workshops.

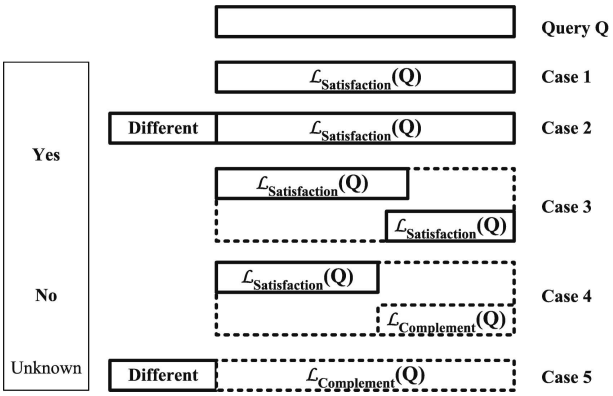

Figure 2: The cases of satisfaction

One should notice that cases 4 and 5 would conduct to a failure of the query when only one mediator is implied. But, if we assume a grouping of mediators (into a federation of mediators), these cases are typical cases where cooperation among the mediators is required. In the case 5, the whole query is transmitted for evaluation to other mediators whereas in the case 4, we need to determine "what is missing" to the individuals to satisfy Q, that means to determine what part of the query is not satisfied by the found individuals. This part as well as the original query are transmitted then to a mediator of the federation. Conceptually, we can see the query as being addressed to "the union" of by the federated mediators' knowledge bases. Concretely, this union is explored from "near to near" within the federation, that means from a mediator to an other.

Let us now elaborate more on the conceptual framework and the underlying formal foundations. Given a query Q expressed as a sentence of a query language $\mathcal{L}_{Query}$, a simple result (noted $\mathcal{L}_{Answer}(Q)$) is generally returned in most of the systems and usually $\mathcal{L}_{Answer}(Q) \doteq$\{Yes, No, Unknown\}, the meaning of "Yes" is that one or several entities satisfy the $\mathcal{L}_{Query}$ and "No" is the opposite. In this work, we try to offer a $\mathcal{L}_{Answer}(Q)$ that may not be a single entity satisfying $\mathcal{L}_{Query}$, but that may possibly be a collection of entities where "the union" of the members of the collection of entities satisfies the search criteria of $\mathcal{L}_{Query}$. This collection can be found in a knowledge base or in a federation of knowledge bases (that means a set of knowledge bases).

In this work, we adopted a Description Logic language (DL) (DL-org, 2003), that were intensively developed and studied in the field of Knowledge Representation. So it is not surprising that they are particularly adapted for representing the semantics of real world situations including data semantics (Calvanese et al., 1998; Borgida, 1995). The query language is

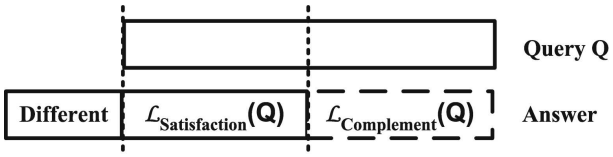

Figure 3: A model of composite answer

defined by $\mathcal{L}_{Query} \doteq$\{ DLs formulaes\}, and $\mathcal{L}_{Answer}$ is defined by two components the $\mathcal{L}_{Satisfaction}$ and $\mathcal{L}_{Complement}$. $\mathcal{L}_{Satisfaction}$ describes a satisfaction, that is a single entity or a set of entities satisfying a query Q. If Q is completely satisfied, its complement (noted $\mathcal{L}_{Complement}(Q)$) will be empty. In the contrary, the system will try to determine a complement for this answer. We define a function $Comp(-,-)$ to calculate the complement of an answer to a query: $\mathcal{L}_{Complement}(Q) = Comp(\mathcal{L}_{Satisfaction}(Q), Q)$. Intuitively this complement designates "the missing part" to an entity in order for that entity to satisfy the query.

The ultimate goal of this work is the design and the development of a set of services to export, import and mediate, as well as the study and the development of alternative strategies and mechanisms for mediators cooperation. The coming sections detail the adopted approach and its foundations, beginning with a short introduction to description logics.

3 COMPLEMENT AND COMPOSITE ANSWER

The determination of unsatisfied part of a query is founded on the concept of *complement* (Schmidt-Schauss and Smolka, 1991) in DLs and the test of the relation of subsumption between the concepts of the query and those in the mediator's base. The algorithm that we propose to test if this relation exists or not, presents the originality, in the case where the relation is not true, to identify the concepts of the query that are the reason of the non existence of the relation (concretely, it is about the concepts of the query that are not subsumed by the concepts in the mediator's base): these concepts describe "the part that is missing" to the individuals who partly satisfy the query. Let us now describe the proposal in a more formal way, beginning by defining the notion of complement to arrive progressively to the procedure of its calculation.

3.1 An introduction to DLs

DLs (DL-org, 2003; Horrocks, 2002b; Napoli, 1997) is a family of knowledge representation languages

where a description of a world is built using *concepts*, *roles* and *individuals*. The *concepts* model classes (sets of *concepts*, TBox)of individuals (sets of *individuals*, ABox) and they correspond to generic entities in an application domain. An *individual* is an instance of a concept. *Roles* model binary relationships among the individual classes. A concept is specified thanks to a structured description that is built giving *constructors* that introduce the roles associated with the concept and possible restrictions associated with some roles. Usually, the restrictions constrain the range of the binary relationship that is defined by a role and the role's cardinality.

PERSON $\dot{\sqsubseteq}$	TOP
SET $\dot{\sqsubseteq}$	TOP
MAN $\dot{\sqsubseteq}$	PERSON
WOMAN $\dot{\sqsubseteq}$	(and PERSON (not MAN))
member $\dot{\sqsubseteq}$	toprole
chef $\dot{\sqsubseteq}$	member
TEAM $\doteq$	(and SET (all member PERSON) (atleast 2 member))
SMALL-TEAM $\doteq$	(and TEAM (atmost 5 member))
MODERN-TEAM $\doteq$	(and TEAM (atmost 4 member) (atleast 1 chef) (all chef WOMAN))

Figure 4: An exemple of LDs TBox

Concepts are of two types, primitive and defined concepts. *Primitive concepts* may be considered as atoms that may serve to build new concepts (the *defined concepts*). Similarly, *roles* may be primitive roles as well as defined roles. In the figure 4, PERSON and SET are primitive concepts: they are introduced using the symbol $\dot{\sqsubseteq}$ and they are linked to a "TOP" concept ($\top$)[2]; TEAM SMALL-TEAM and MODERN-TEAM are defined concepts (they are introduced using the symbol $\doteq$. The *and* constructor enables defining concepts as a conjunction of concepts: these concepts are the immediate ascendants of the defined one. The *all* constructor constrains a role's range and the *at-least* and *at-most* constructors enable specifying the role's cardinals. Finally, the *not* constructor only applies to primitive concept.

Subsumption is the fundamental relationship that may hold among described concepts. Intuitively, a concept C subsumes a concept D, if the set of individuals represented by C contains the set of individuals represented by D. More formally, C subsumes D and it is denoted as $D \sqsubseteq C$ (or D is subsumed by C) if and only if $D^{\mathcal{I}}$ for every possible interpretation $\mathcal{I}$. C is called the subsuming concept and D is the subsumed. For example in figure 4, PERSON subsumes MAN, SMALL-TEAM is subsumed by TEAM.

The subsumption relationship organizes the concepts into a hierarchy of concepts. The classification process aims at determining the position of a new concept in a given hierarchy. In this framework, a query is represented as a concept Q to be classified in a given hierarchy. The result of the query is the set of instances of the concepts that are subsumed by Q. The classification is a process that enables discovering whether a subsumption relationship holds between a concept X, and those present in a hierarchy H. The classification process is decomposed into 3 steps:

- Retrieve the most specific subsuming concepts of X (denoted MSSC(X));
- Retrieve the most general concepts subsumed by X (denoted MGSC(X));
- (possibly) Update the links between X and its MSSC and its MGSC.

The result of MSSC and MGSC can determine the cases of satisfaction (see figure 5).

In DLs a classic definition of the complement[3]is: given two descriptions A and B in $\mathcal{ALCNR}$[3] as $A \sqsubseteq B$, the complementary of A relatively to B, noted $Comp(A, B)$, is defined as the description C is most general as $A \equiv B \sqcap C$. $Comp(A, B)$ corresponds to the minimal additional knowledge that is missing to a process of B to be an instance of A. For example, Comp(SMALL-TEAM, TEAM) = (at-most 5 member). The complementary of a concept relatively to another concept and smallest subsuming common of two concepts. This definition of the complement requires $(B \sqcap C) \equiv A$, where $(B \sqcap C) \equiv A \Leftrightarrow ((B \sqcap C) \sqsubseteq A) \wedge ((B \sqcap C) \sqsupseteq A)$. The determination of a complement is based on the subsumption relationship as explained in the next section where the test of the existence of the subsumption relationship is based on a Normalize-Compare process.

3.2 Determination of the Complement Concept

The aim of a normalization process is to put defined concepts to be compared, let say A and B, under a conjunctive form: $A = (\text{ and } A_1, A_2, A_3, \ldots, A_n)$ and $B = (\text{ and } B_1, B_2, B_3, \ldots, B_m)$. Once

[2]Intuitively the TOP concept is the "most general one" and it contains all the individuals while the BOTTOM concept ($\bot$) is the most specific one and is empty.

[3]$\mathcal{ALCNR}$ is a family of representation langage of DLs

normalized, two concepts can be easily compared to check wither the subsumption relationship holds between pairs of them or not: giving $A = ($ and $A_1, A_2, A_3, \ldots, A_n)$ and $B = ($ and $B_1, B_2, B_3, \ldots, B_m)$, the test "does the concept A subsume the concept B?" returns "true", if and only if $\forall A_i$ (i $\in$ 1,...,n) $\exists$ B_j (j $\in$ 1,...,m) such that $B_j \sqsubseteq A_i$.

The implementation of this process uses an array of Boolean (called "Table_Of_Test" further) to record the result of the subsumption relationship evaluation (see figure 5). In that figure, $C_1, C_2, C_3, \cdots, C_n$ denote the query concept under its normal form and $D_1, D_2, D_3, \ldots, D_m$ denotes the concepts "known from" the mediators, i.e. every D_j has to be viewed under its normal form $D_j^1, D_j^2, \cdots, D_j^{nj}$. Then $Table\_Of\_Test[D_j, C_i] = true$ means that $D_j^i \sqsubseteq C_i$. When the value returned by the function $Subsumes(C, D_j)$ is "false" (i.e. the concept D_j does not fully satisfy the concept C.), therefore we need to determine a possible complement of D_j relatively to C. Using the Table_Of_Test it is easy to get the complement of the concept D_j relatively to the concept C. For example, "MODERN-TEAM subsumes SMALL-TEAM" is false, "(atmost 4 member), (atleast 1 chef) and (all chef WOMAN)" is false, then $Comp$(MODERN-TEAM,SMALL-TEAM)=(and (atmost 4 member) (atleast 1 chef) (all chef WOMAN)). Then using Table_Of_Test the definition of $Comp()$ is: $Comp(C, D) = \bigcap_{k=1}^{r} C_k, \forall k | TableOfTest[k] = false$[4].

	C_1	C_2	...	C_n
D_1	False	False	...	True
D_2	False	True	...	True
...	...	...	...	...
D_m	False	False	...	False
ORoD	False	True	...	True

ORoS	ANDoS
True	False

Figure 5: Three parameters of "Table Of Test"

That means that the complement is given by the conjunction of all the atomic concepts for which the corresponding values in "Table Of Test" are "false".

The composition of the truth values will permit us to determine the cases of satisfaction. Let's consider a table $ORoD[1..n]$ as $ORoD[i] = \bigvee_{j=1}^{m} T[D_j, C_i]$. $ORoD[i] = true$ means that the concept C_i is satisfied by at least a D_k. If the conjunction of the values of $ORoD$, noted $ANDoS$, is true (i.e. $\bigwedge_{i=1}^{n} ORoD[i] = True$), it means that all the C_is are satisfied and therefore the query. When $ANDoS$ is false, the logical disjunction of the values of $ORoD$, noted $ORoS$, enables to determine a possible partial satisfaction. Indeed if $ORoS = \bigvee_{i=1}^{n} ORoD[i] = True$, it means that there exist some C_k that are satisfied. If both $ORoS$ and $ANDoS$ are false then no atomic concept D_j^k ($j \in 1..m$) satisfies a C_i.

MSSC	MGSC	ORoS	ANDoS	CASE
X	X	True	True	1
X	$\bot$	True	True	2
$\top$	$\bot$	True	True	3
$\top$	$\bot$	True	False	4
$\top$	$\bot$	False	False	5

Figure 6: The analyse of the satisfaction case

The figure 6[5] summarizes this discussion (in this figure, the numbers in the CASE column refers to the satisfaction cases listed in section 2). By this analyse with the classification algorithm (Boudjlida, 2002), we can get a complete method to determine the satisfaction cases that were listed in section 2.

3.3 Status of The Implementation

Based on these algorithms, an experimental platform has been developed in *Java*. Some services, like testing the subsomption relationship, determining the complement of a concept and computing a composite answer, have been implemented on this platform. All these services can be accessed through a *Web Server* in a "classical" client/server architecture over the Web. The services accept the DL concepts written DAML+OIL (Horrocks, 2002a), an ontology language in *XML*.

The DL concepts are encoded in *XML* and transmitted to the *Web Server* who in turn transmit them to the appropriate mediator service. Then a normalization class transforms them into *Java* objects and an experimental knowledge base, described in XML, is loaded and normalized into a *TBox* object when the service is started. All the algorithms of the mediator's services are implemented in the *TBox* class. The services' outputs are also encoded in *XML*. Furhermore, XML is also used to exchange information and concepts between the mediators when mediators' copperation is needed. For example, in the service that computes a composite answer for a query concept, if the query concept is not satisfied in the local knowledge base, the complement concept is sent to next mediator as

[4] In the following, we will use T[x] instead of TableOfTest[x]

[5] In this table, X is a concept, $\top$ is the concept TOP and $\bot$ is the concept BOTTOM.

an XML focument.

In the current status of the implementation, a mediator "discovers" an other mediator using a static mediator address list. More complex and dynamic discovery techniques will be supported in the coming versions. Moreover, we deliberately ignored the search of the actual *individuals* (*ABox*) that satisfy a query, i.e. in the current work, we only consider *TBoxes*.

4 CONCLUSION

In this paper, we presented a method and an algorithm for testing the subsumption relationship, determining concept complements and finding composite answers. Some service of subsumption test and determine *Complement* has been implemented in *Java*, that is based on a normalization-comparison algorithm and we can access these services by Web. One of the originality of this work is in the type of query answering we provide and also in the way we used and implemented the complement concept. Indeed, to the best of our knowledge, using the complement concept for query answers composition does not figure in the literature we had in hands nor in systems that implement DLs, like *RACER* (Haaslev and Moller, 2003) one of the most representative DL system.

Future work may consider the complexity of the algorithm, and the cooperation of mediators in heterogeneous environments, i.e. environments where mediators' knowledge bases are described in different languages.

REFERENCES

Beeri, C., Levy, A., and Rousset, M.-C. (1997). Rewriting Queries Using Views in Description Logics. In *ACM Symposium on Principles Of Database Systems*, pages 99–108, Tucson, Arizona.

Borgida, A. (1995). Description Logics in Data Management. *IEEE Transactions on Knowledge and Data Engineering*, 7(5):671–682.

Borgida, A. and Devanhu, P. (1999). Adding more "DL" to IDL: Towards more Knowledgeable Component Interoperability. In *21rst International Conference on Software Engineering, ICSE'99*, pages 378–387, Los Angeles, CA. ACM Press.

Bouchikhi, M. and Boudjlida, N. (1998). Using Larch to Specify the Behavior of Objects in Open Distributed Environments. In *Proceedings of the 1998 Maghrebian Conference on Software Engineering and Artificial Intelligence*, pages 275–287, Tunis, Tunisia. 98-R-300.

Boudjlida, N. (1995). Knowledge in Interoperable and Evolutionary Systems. In Dreschler-Fischer, L. and Pribbenow, S., editors, *KRDB'95, Workshop on "Reasoning about Structured Objets: Knowledge Representation Meets Databases"*, pages 25–26, Bielefeld, Germany. (Position Paper).

Boudjlida, N. (2002). A Mediator-Based Architecture for Capability Management. In Hamza, M., editor, *Proceedings of the 6th International Conference on Software Engineering and Applications, SEA 2002*, pages 45–50, MIT, Cambridge, MA.

Calvanese, D., de Giacomo, D., Lenzerini, M., Nardi, D., and Rosati, R. (1998). Information Integration: Coceptual Modeling and Reasoning Support. In *6th International Conference on Cooperative Information Systems, CoopIS'98*, pages 280–291.

DL-org (2003). Description logics. http//dl.kr.org/.

Dyck, T. (2002). Uddi 2.0 provides ties that bind. (http://www.eweek.com/).

Haaslev, V. and Moller, R. (2003). Racer: Renamed abox and concept expression reasoner. http://www.fh-wedel.de/mo/racer/index.html.

Han, T.-D., Purao, S., and Storey, V. (1999). A Methodology for Building a Repository of Object-Oriented Design Fragments. In *18th International Conference on Conceptual Modelling, ER'99*, pages 203–217, Paris. Spriger Verlag. LNCS 1728.

Horrocks, I. (2002a). DAML+OIL: a description logic for the semantic web. *IEEE Data Engineering Bulletin*, 25(1):4–9.

Horrocks, I. (2002b). Description Logic: Axioms and Rules. Talk given at Dagstuhl "Rule Markup Technique" Workshop. http://www.cs.man.ac.uk/ horrocks/Slides/.

Napoli, A. (1997). Une introduction aux logiques de description. Technical Report RR No 3314, INRIA-LORIA, Nancy.

Nonaka, I. and Takeuchi, H. (1995). *The Knowledge Creating Company; How Japanese Companies Create the Dynamics of Innovation*. Oxford University Press.

OMG (1992). The Object Model Architecture Guide. Technical Report 91.11.1, Revision 2.0, Object Management Group.

Schmidt-Schauss, M. and Smolka, G. (1991). Attribute Concepts Description with Complements. *Artificial Intelligence Journal*, 48(1):1–26.

uddi.org (2000). UDDI: Universal Description, Discovery and Integration. Technical White Paper. (http://uddi.org).

W3C (2003a). Semantic Web. http://www.w3.org/2001/sw.

W3C (2003b). Web Ontology. http://www.w3.org/2001/sw/WebOnt.

W3C (2003c). Web Services. http://www.w3.org/2002/ws.