Semantic Matchmaking in a P-2-P Electronic Marketplace

Tommaso Di Noia, Eugenio Di Sciascio, Francesco M. Donini, Marina Mongiello
Dipartimento di Elettrotecnica ed Elettronica
Politecnico di Bari
Via Re David, 200
70125 BARI, Italy
t.dinoia,disciascio,donini,mongiello@poliba.it

ABSTRACT

Matchmaking is the problem of matching offers and requests, such as supply and demand in a marketplace, services and customers in a service agency, etc., where both partners are peers in the transaction. Peer-to-Peer (P-2-P) e-commerce calls for an infrastructure treating in a uniform way supply and demand, which should base the match on a common ontology for describing both supply and demand. Knowledge representation — in particular description logics — can deal with this uniform treatment of knowledge from vendors and customers, by modelling both as generic concepts to be matched. We propose a logical approach to supply-demand matching in P-2-P e-commerce, which allows us to clearly distinguish between exact, potential and partial match, and to define a ranking within the categories. The approach is deployed in a prototype system implemented for a particular case study (but easily generalizable) and is based on Classic, a well-known knowledge representation system.

Categories and Subject Descriptors

H.4.2 [**Information Systems Applications**]: Decision Support; I.2.4 [**Knowledge Representation Formalisms and Methods**]: Representation languages

Keywords

Matchmaking, knowledge representation, web services, description logics.

1. INTRODUCTION

The process of searching the space of possible matches between demands and supplies can be defined as Matchmaking. This process is, or should be, quite different from simply finding, given a demand, a perfectly matching supply (or vice versa). Instead it includes finding all those supplies that can *to some extent* fulfill a demand, and eventually propose the best ones.

SAC 2003 Melbourne, Florida, USA

Earliest matchmakers, based on KQML, were proposed in [7] and [13]. Similar approaches were deployed in SIMS [1], which used KQML and LOOM as description language and InfoSleuth [11], which adopted KIF and the deductive database language LDL++. LOOM is also at the basis of the subsumption matching addressed in [8]. Recent relevant literature includes [18] and [15]. The matching process is carried out here through five progressive stages, going from classical Information Retrieval (IR) analysis of text to semantic match via Θ-subsumption. The notion, inspired by Software Engineering, of *plug-in* match is introduced to overcome the limitations of a matching approach based on exact match. No ranking is presented but for what is called relaxed match, which basically reverts again to a IR free-text similarity measure. So a basic service of a semantic approach, such as inconsistency check, seems unavailable with this type of match. In [19] and [9] a matchmaking framework is proposed, which operates on service descriptions in DAML+OIL and is based on the FaCT reasoner. Unfortunately FaCT lacks of concrete datatypes, which are obviously extremely useful for e-commerce applications, and their prototype is incomplete. Semantic service discovery via matchmaking in the Bluetooth framework is investigated in [17]. Also here the issue of approximate matches, to be somehow ranked and proposed in the absence of exact matches, is discussed, but as in the previous papers no formal framework is given. Also commercial electronic marketplaces try to provide some matchmaking capabilities between demand and supply. Jango (www.jango.com) provides a system that basically only allows comparison, in terms of price, of goods available in on-line stores on the Internet. Obviously the description of the product to be matched has to be complete and consistent and no reasoning on set containment or inconsistency check can be carried out. PersonaLogic (www.PersonaLogic.com) allows customers to impose constraints for alternatives seeking. It must be pointed out that constraints cannot be dynamically placed but have to be taken from a pre-determined category set. Kasbah (www.kasbah.com) is a more effective system, which allows to dynamically set constraints, yet it does not allow handling of inconsistency and partial or potential matches. A similar approach is also deployed in Tete-a-Tete [14]. An advanced constraint based approach is proposed in [12], which allows to handle conflicting preferences in demands/supplies. Consistency check of preferences is accomplished visiting an offer synthesis graph with path consistency algorithm each time a new offer is entered. Smartclient [16] is also a system that allows users criteria

adjustment using an interface that shows the initial search space —which may result quite large— and then user interaction with the results. The underlying system basically relies on partial constraint satisfaction techniques. A recent proposal along the same lines is in [20] where negotiation agents are formally modelled using an object-oriented constraint language. IBM's Websphere matchmaking environment is, to our knowledge, the first example of commercial solution that places an explicit emphasis on the matchmaking between a demand and a supply in a peer-to-peer way, which is referred to in [10] as *symmetric* matchmaking. The environment is based on a matchmaking engine that describes supplies/demands as properties and rules. Properties are name-value pairs constructed using an extension of Corba Trading service language. Rules are basically constructed using a generic script language. Matching is then accomplished by simply comparing properties and verifying rules. A similar approach is in [4]. In [5] an initial setting for logical matchmaking was presented in a person-to-person framework.

As a general consideration, we note that if supplies and demands are simple names or strings, the only possible match would be identity, resulting in an all-or-nothing approach to matchmaking. Although effective for fixed technical domains, *e.g.*, Bluetooth use of UUID descriptors, such an approach misses the fact that supplies and demands usually have some sort of structure in them. Such a structure could be exploited in order to evaluate "interesting" *inexact* matches. Vector-based techniques taken by classical IR can be used, too, thus reverting matchmaking to similarity between weighted vectors of stemmed terms, as proposed in the COINS matchmaker [13] or in LARKS [18]. Obviously lack of document structure in descriptions would make matching only probabilistic and strange situations may ensue, *e.g.*, consider a simple demand *"apartment with two Rooms in Soho pets allowed no smokers"* and a supply *"apartment with two Rooms in Soho, no pets, smokers allowed"*. They would correspond to a perfect match although being in obvious contrast.

We propose a matchmaking framework that allows to logically distinguish and classify:

Total match: all requests in Demand are available in Supply (or vice versa)

Potential match: some requests in Demand are not specified in Supply (and further action -inquire- can be taken)

Partial match: some requests in Demand are in contrast with Supply (and further action -retract- can be taken)

This coarse subdivision allows to immediately classify a given supply/demand description in a category. As a further step beyond we propose an algorithm to rank each description within its own category. This is of extreme importance for a practical use of the approach, as the key question that has to be answered is how far is a given demand (supply) from its counterpart? And which are the requirements that would eventually fulfill it?

In the remaining of this paper we concentrate on matchmaking in a P-2-P scenario, setting a logical framework for it, and highlighting general properties that should hold for a matchmaking algorithm, and for the associated ranking functions. We then apply our framework to DL for expressing offers and requests, adapting the general principles to a DL knowledge base. Finally, we describe a matchmaking facilitator prototype system, which is inspired by the principles discussed so far.

2. PRELIMINARY NOTIONS

Description Logics (DLs) are a family of logic formalisms for Knowledge Representation. Here we give some basics about DLs, but for space bounds we limit to the DL of the KR system CLASSIC that we use in the paper. CLASSIC [2] has been developed at AT&T Bell Labs, where it has been applied in several projects about configuration and program repositories. Its language has been designed with the goal to be as expressive as possible while still admitting polynomial-time inferences.

The basic syntax elements are *concept* names, *e.g.*, `book`, `person`, `product`, `apartment`, and *role* names, like `author`, `supplier`, `hasRooms`. Intuitively, concepts stand for sets of objects, and roles link objects in different concepts, *e.g.*, the role `author` links books to persons (their writers). Formally, an *interpretation* is defined as a pair $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$, which consists of the *domain* Δ and the *interpretation function* $\cdot^{\mathcal{I}}$, which maps every concept to a subset of Δ, and every role to a subset of $\Delta \times \Delta$.

Concept and role names can be combined using *constructors* to form concept and role *expressions*, and each DL has its distinguished set of constructors. First of all, every DL allows one to form a *conjunction* of concepts, usually denoted as $\sqcap$. Roles can be combined with concepts using *universal role quantification*, as in $\texttt{product} \sqcap \forall \texttt{supplier.japanese}$, which describes products sold only by japanese suppliers. Other constructs may involve counting, as number restrictions: $\texttt{apartment} \sqcap (\leq\ 1\ \texttt{hasRooms})$ expresses apartments with just one room, and $\texttt{book} \sqcap (\geq\ 3\ \texttt{author})$ describes books written by at least three people. CLASSIC provides also other constructs, such as `same-as`, `fills`, and `one-of`. We omit their presentation, since we do not make use of them in this paper.

Expressions are given a semantics by defining the interpretation function over each construct. Concept conjunction is interpreted as set intersection: $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$ The interpretation of constructs involving quantification on roles needs to make domain elements explicit: for example, $(\forall R.C)^{\mathcal{I}} = \{d_1 \in \Delta \mid \forall d_2 \in \Delta : (d_1, d_2) \in R^{\mathcal{I}} \rightarrow\ d_2 \in C^{\mathcal{I}}\}$.

Concept expressions can be used in *inclusion assertions*, and *definitions*, which introduce the meaning of concept names in terms of concept expressions. Definitions give a meaningful name to particular combinations of concepts, as in $\texttt{doubleRoom} \equiv \texttt{room} \sqcap (=\ 2\ \texttt{hasPlaces})$. Inclusions can be used in a similar way, but they state only *necessary conditions* for the concept name on the left, as in $\texttt{book} \sqsubseteq (\geq\ 1\ \texttt{author}) \sqcap (=\ 1\ \texttt{title}) \sqcap (=\ 1\ \texttt{ISBN})$. Intuitively, every book has at least one author and exactly one title, but not everything having at least one author and exactly one title is a book (*e.g.*, a theater performance). Another form of assertions that are present in CLASSIC are disjoint groups of names, *e.g.*, `paperbacks` and `hardcover`. In more expressive DL, such assertions are modeled as general inclusions. Historically, the set of all such inclusions and definitions in a DL knowledge base is called TBox (Terminological Box). As for semantics, inclusions and definitions impose restrictions on possible interpretations: an interpretation $\mathcal{I}$ satisfies an inclusion $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and it satisfies a definition $C = D$ when $C^{\mathcal{I}} = D^{\mathcal{I}}$. A *model* of a TBox T is an interpretation that satisfies all inclusions and definitions of T.

It is important to note that every CLASSIC C concept has

an equivalent normal form as $C_{names} \sqcap C_\sharp \sqcap C_{all}$, in which C_{names} is a conjunction of names, $C_\sharp$ of number restrictions, and C_{all} of universal role quantifications. In the normal form, also all inclusions, definitions and disjoint groups have been made explicit [3].

CLASSIC provides the two basic reasoning services of DL-based systems, namely *Concept Satisfiability* (given a TBox T and a concept C, does there exist at least one model of T assigning a non-empty extension to C?), and *Subsumption* (given a TBox T and two concepts C and D, is C more general than D in any model of T?). Being a complete KR system, CLASSIC provides also data types as numbers and strings, and other services which are useful in a deployed prototype.

3. KR FOR MATCHMAKING

We now discuss some general principles that an approach to matchmaking based on Knowledge Representation (KR) may yield.

First of all, a matchmaking facilitator o practical use should be liberal enough about details, without pretending a proposer to fill in forms with (say) 30 or more different characteristics to be set. This implies that the absence of a characteristic in the description of a supply or demand should not be interpreted as a constraint of absence. Instead, it should be considered as a characteristic that could be either refined later, or left open if it is irrelevant for a user — what is called *open-world assumption* in KR. Obviously, also the algorithm employed for matchmaking should take this issue into account. Secondly, a matchmaking system may give different evaluations depending on whether it is trying to match a supply S with a demand D, or D with S — *i.e.*, depending on *who* is going to use this evaluation. This requirement is already evident when characteristics are modeled as sets of words: in that case, underconstrained requirements of S from the point of view of D are expressed by $D - S$ (set difference) while underconstrained requirements of D from S's viewpoint are expressed as $S - D$.

Of course, using sets of words to model supplies and demands would be too sensible to the choice of words employed — it misses meanings that relate words. In the apartments rental scenario, a matching facilitator should take into account that "boiler" is a form of heating system, or that a constraint "no-pets" applies also to a dog. It is now a common opinion that such fixed-terminology problems are overcome if terms have a logical meaning through an ontology [6].

From now on we suppose that supplies and demands are expressed in a description logic. We note that this approach includes the sets-of-keywords one, since a set of keywords can be considered also as a conjunction of concept names, *e.g.*, the set $\{apartment, soho, twoRooms\}$ can be equivalently considered as $\texttt{apartment} \sqcap \texttt{soho} \sqcap \texttt{twoRooms}$ — without modelling the structure of concepts. Of course, entering into the structure of concepts may yield $\texttt{apartment} \sqcap \forall \texttt{location.soho} \sqcap (= \ 2\ \texttt{hasRooms})$, while a keyword "noPets" can be given a logical meaning as $\forall \texttt{occupants.}(\leq \ 0\ \texttt{hasPets})$, which is probably better.

We suppose also that a common ontology for supplies and demands is established, as a TBox in DL. Now a match between a supply S and a demand D could be evaluated according to T. Let $T \models \ldots$ denote logical implication (truth in all models of T), and let $\sqsubseteq$ (subsumption) denote also implication between constraints of S and D. There are three relations between concepts expressing supplies and demands, that we consider meaningful in matchmaking:

Implication. If $T \models (D \sqsubseteq S)$, then every constraint imposed by D is fulfilled (implied) by S, and vice versa if $T \models (S \sqsubseteq D)$.

Consistency. If $D \sqcap S$ is satisfiable in T, then there is a *potential* match, in the sense that the constraints of neither proposal exclude the other. This relation has been highlighted also by other researchers [19].

Inconsistency. Otherwise, if $D \sqcap S$ is unsatisfiable in T, some constraints of one proposal are in contrast with the properties of the other one. However, also (say) supplies which are inconsistent with D may be reconsidered, if the demander accepts to *revise* some of D's constraints. We call this situation a *near miss* or *partial match*.

We now state some properties that — we believe — every ranking function should have in logical matchmaking. We state these properties in form of definitions, since we distinguish between rankings having the property from rankings that do not.

First of all, a ranking for semantic matchmaking should be *syntax independent*. That is, for every pair of supplies S_1 and S_2, demand D, and ontology T, when S_1 is logically equivalent to S_2 then S_1 and S_2 should have the same ranking for D — and the same should hold also for every pair of logically equivalent demands D_1, D_2 with respect to every supply S. For example, an apartment S_1, described as available for the summer quarter, should have the same rank — with respect to a request — as another S_2, identical but for the fact that it is described to be available for June-July-August. A similar property should hold also for ranking incoherent pairs of supplies and demands (see the full paper).

Secondly, a ranking for semantic matchmaking should be *monotonic over subsumption*. That is, for every demand D, for every pair of supplies S_1 and S_2, and ontology T, if S_1 and S_2 are both potential matches for D, and $T \models (S_2 \sqsubseteq S_1)$, then S_2 should be ranked either the same, or better than S_1. The same should hold also for every pair of demands D_1, D_2 with respect to a supply S. Intuitively, this property could be read of as "A ranking of potential matches is monotonic over subsumption if the more specific, the better." Observe that we use the word "better" instead of using any symbol $\leq, \geq$. This is because some rankings may assume that "better=increasing" (towards infinity, or 1) while others may assume "better=decreasing" (towards 0).

When turning to partial matches, adding another characteristic to an unsatisfactory proposal may either worsen its ranking (when another characteristic is violated) or keep it the same (when the new characteristic is not in contrast). Note that this ranking should be kept different from the ranking for potential matches.

We remark that the properties we stated in this section are independent of the particular DL employed, or even the particular *logic* chosen. For instance, the same properties could be stated if propositional logic was used to describe supplies, demands and the ontology. In this respect, we believe that the properties above keep a general significance.

4. THE MATCHMAKING ALGORITHM

Algorithms for potential and partial matchmaking have been devised adapting the original CLASSIC structural algorithm for subsumption [3]. In this section we thoroughly describe only the algorithm for potential match; the algorithm for partial match is similar to the potential one, and is only briefly described here for space reasons.

A CLASSIC concept C can be put in normal form as $C_{names} \sqcap C_\sharp \sqcap C_{all}$. Without ambiguity, we use the three components also as sets of the conjoined concepts. Moreover, recall that the TBox in CLASSIC can be embedded into the concepts, hence we do not consider explicitly the TBox, although it is present.

The algorithm easily follows a structural subsumption algorithm, except for the treatment of universal role quantification, that we explain now with the help of an example. Suppose that D is a demand and C_1, C_2 are two supplies defined as follows:

$$D = \texttt{apartment} \sqcap \forall \texttt{hasRooms.}(\texttt{singleRoom} \sqcap \texttt{nonSmokerRoom})$$
$$C_1 = \texttt{apartment} \sqcap \forall \texttt{hasRooms.roomWithTV}$$
$$C_2 = \texttt{apartment}$$

Now, comparing D with C_1, a recursive call through the universal role quantification highlights that rooms in C_1 miss both characteristics required by D, hence the ranking should be 2 (where ranking 0 would mean subsumption). In the case of C_2, instead, since the universal role quantification is absent, no recursive comparison is possible. However, observe that from the semantics, $\forall \texttt{hasRooms.} \top \equiv \top$ (no restriction on the fillers of role `hasRooms` is equivalent to no restrictions at all). Hence, $\texttt{apartment} \equiv (\texttt{apartment} \sqcap \forall \texttt{hasRooms.} \top)$. Since we want to enforce syntax independence, both concepts should yield the same ranking. Hence, we compare the last concept, which allows us to make a recursive comparison of characteristics of universal role quantifications.

Algorithm $rankPotential(C, D)$;
input CLASSIC concepts C, D, in normal form, such that $C \sqcap D$ is satisfiable
output rank $n \geq 0$ of C w.r.t. D, where 0 means that $C \sqsubseteq D$ (best ranking)
begin algorithm
  let $n := 0$ **in**
    /* add to n the number of concept names in D */
    /* which are not among the concept names of C */
    1. $n := n + |D_{names+} - C_{names+}|$;
    /* add to n number restrictions of D */
    /* which are not implied by those of C */
    2. **for each** concept $(\geq\ x\ R) \in D_\sharp$
    such that there is no concept $(\geq\ y\ R) \in C_\sharp$ with $y \geq x$
      $n := n + 1$;
    3. **for each** concept $(\leq\ x\ R) \in D_\sharp$
    such that there is no concept $(\leq\ y\ R) \in C_\sharp$ with $y \leq x$
      $n := n + 1$;
    /* for each universal role quantification in D */
    /* add the result of a recursive call */
    4. **for each** concept $\forall R.E \in D_{all}$
      ifthere does not exist $\forall R.F \in C_{all}$
        then $n := n + rankPotential(\top, E)$;
        else $n := n + rankPotential(F, E)$;
    return n;
end algorithm

Obviously, total match is a particular case of potential match, obtained when $rankPotential(C, D) = 0$.

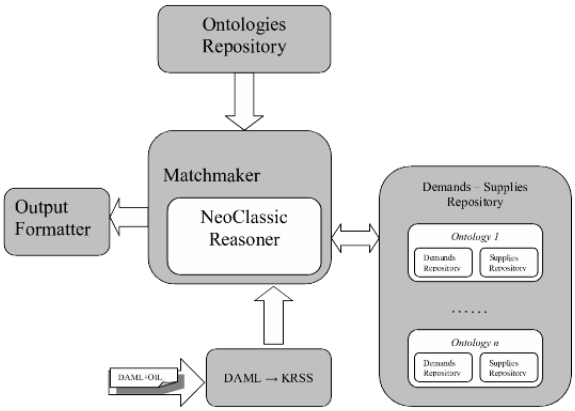

Figure 1: The architecture of the engine

It is easy to modify the algorithm if weights on subconcepts of D are taken into account: instead of adding 1 to n for each D's concept missing in C, one just adds the corresponding weight. In this way, when the proposal concerns apartments, the concept `apartment` gets the highest weight, and minor characteristics get lower weights. Then, a far rank would mean that either many minor characteristic, or a very important one, are left unspecified in C.

We implemented also a version of the algorithm in which weights are *learned* by the system, upon repeated analysis of proposals. In this case, of course, the learned weights are *absolute* ones, and not relative to a particular actor. However, also in this case, if all proposals contain *e.g.*, the concept `apartment`, this concept gains weight, which is reasonable if the ontology refers to real estate.

The algorithm for ranking partial matches follows again the partition of CLASSIC concepts into names, number restrictions, and universal role quantifications. However, this time we are looking for inconsistencies. Hence, when a universal role quantification is missing in either concept, the recursive call is unnecessary.

Of course, also in this case weights could be added to subconcepts of D, where the greater the weight, the more that characteristic is important, making the rank of C far off when in contrast.

For both algorithms it can be proved they respect the properties highlighted in the previous section.

5. THE MATCHMAKING SYSTEM

The matchmaking framework presented in the previous sections has been deployed in a prototype facilitator. The system embeds a NeoClassic engine (a C++ implementation of the original Classic system, whose sources have been modified for our purposes).

Our matching engine, whose architecture is shown in Figure 1, is based on Java servlets; it embeds the NeoClassic reasoner and communicates with the reasoner running as a background daemon. At this stage of the work, the system is not fully transactional, so requests have to be serialized by the engine.

The system receives a KRSS string describing the demand/supply and the URI referencing the proper ontology. The Reasoner checks the description for consistency; if it fails, based on the reasoner output, the system provides an

error message stating the error occurred. Otherwise the proper matchmaking process takes place. The NeoClassic standard [3] subsumption algorithm has been modified in accordance with the matchmaking algorithms presented in section 4. Each match can return a 0, which means total match or a value > 0. Recall that returned values for partial matches and potential matches have logically different meaning and matching descriptions are sorted in different sets. The matching engine may return then three separate result sets.

Main services our system currently provides are:

Support to the user in the data insertion and query submission. The user is incrementally guided in the definition of a proposal.

Automatic construction and verification of consistency w.r.t. the reference ontology of the demand/supply.

Deduction of new knowledge on the basis of available data.

Ability to provide ranked conceptually approximate answers, *i.e.*, near miss or partial match, in the presence of unsatisfiable queries. Notice that this the way a human clerk behaves: when a request cannot be satisfied he/she will propose the nearest miss to the client; he/she will not answer "no match".

Ability to provide ranked potential matches and possibility to ask for unforeseen (hence not immediately available) services and features to the supplier, with successive automatic update of description and communication of update.

Storage of satisfiable demands that were still unmatched, with automatic reexamination when new supplies are provided, and notification on successful match between supply and demand. The same service is available for unmatched supplies.

We have actually registered the matchmaking web service and we are also in the process of embedding the matchmaking service in the BlueTooth environment.

6. CONCLUSION

In this paper we have set a logical framework for semantic matchmaking, and pointed out general properties that we believe every matchmaking facilitator should have.

In accordance with the above properties we have devised algorithms for matchmaking that allow the categorization of match type and the ranking of matches.

The general framework has been implemented in a matchmaking facilitator. The core engine has been implemented modifying the NeoClassic reasoner for our purposes, and we have shown that, although with a reduced expressiveness w.r.t. more recent reasoners, Classic can be effectively used for this class of problems.

7. ACKNOWLEDGMENTS

We thank Peter Patel-Schneider for valuable support on Classic. This work has been supported by project MURST-CLUSTER22, by EU-POP project *"Negotiation Agents for the Electronic Marktplace"*, by Italian CNR projects LAICO, DeMAnD, and "Metodi di Ragionamento Automatico nella modellazione ed analisi di dominio".

8. REFERENCES

[1] Y. Arens, C. A. Knoblock, and W. Shen. Query Reformulation for Dynamic Information Integration. *J. of Intelligent Information Systems*, 6:99–130, 1996.

[2] A. Borgida et al. CLASSIC: A Structural Data Model for Objects. In *Proc. of ACM SIGMOD*, pages 59–67, 1989.

[3] A. Borgida and P. F. Patel-Schneider. A Semantics and Complete Algorithm for Subsumption in the CLASSIC Description Logic. *JAIR*, 1:277–308, 1994.

[4] F. Casati and M. C. Shan. Dynamic and Adaptive Composition of E-Services. *Information Systems*, 26:143–163, 2001.

[5] E. Di Sciascio et al. A Knowledge-Based System for Person-to-Person E-Commerce. In *Proc. Workshop on Applications of Description Logics (KI 2001)*, 2001.

[6] D. Fensel et al. OIL: An Ontology Infrastructure for the Semantic Web. *IEEE Intelligent Systems*, 16(2):38–45, 2001.

[7] T. Finin et al. KQML as an Agent Communication Language. In *Proc. of CIKM'94*, pages 456–463. ACM, 1994.

[8] Y. Gil and S. Ramachandran. PHOSPHORUS: a Task based Agent Matchmaker. In *Proc. AGENTS '01*, pages 110–111. ACM, 2001.

[9] J. Gonzales-Castillo, D. Trastour, and C. Bartolini. Description Logics for Matchmaking of Services. In *Proc. of Workshop on Application of Description Logics (KI 2001)*, 2001.

[10] Y. Hoffner et al. Distribution Issues in the Design and Implementation of a Virtual Market Place. *Computer Networks*, 32:717–730, 2000.

[11] N. Jacobs and R. Shea. Carnot and Infosleuth – Database Technology and the Web. In *Proc. of ACM SIGMOD*, pages 443–444. ACM, 1995.

[12] N. Karacapilidis and P. Moraitis. Building an Agent-Mediated Electronic Commerce System with Decision Analysis Features. *Decision Support Systems*, 32:53–69, 2001.

[13] D. Kuokka and L. Harada. Integrating Information Via Matchmaking. *J. of Intelligent Information Systems*, 6:261–279, 1996.

[14] P. Maes, R. Guttman, and A. Moukas. Agents that Buy and Sell. *Comm. of the ACM*, 42(3):81–91, 1999.

[15] M. Paolucci et al. Semantic Matching of Web Services Capabilities. In *The Semantic Web - ISWC 2002*, number 2342 in LNCS, pages 333–347. Springer-Verlag, 2002.

[16] P. Pu and B. Faltings. Enriching Buyers' Experience. *CHI Letters*, 2(1), 2000.

[17] S.Avancha, A. Joshi, and T. Finin. Enhanced Service Discovery in Bluetooth. *IEEE Computer*, pages 96–99, 2002.

[18] K. Sycara et al. LARKS: Dynamic Matchmaking Among Heterogeneus Ssoftware Agents in Cyberspace. *Autonomous agents and multi-agent systems*, 5:173–203, 2002.

[19] D. Trastour, C. Bartolini, and C. Priest. Semantic Web Support for the Business-to-Business E-Commerce Lifecycle. In *Proc. WWW '02*, pages 89–98. ACM, 2002.

[20] H. Wang, S. Liao, and L. Liao. Modeling Constraint-Based Negotiating Agents. *Decision Support Systems*, 33:201–217, 2002.