A SURVEY ON WEB SERVICE DISCOVERING AND COMPOSITION

Elena del Val Noguera, Miguel Rebollo Pedruelo
Department of Information Systems and Computation
Technical University of Valencia
Camino de Vera s/n, Valencia, Spain
{edelval,mrebollo}@dsic.upv.es

Keywords: Web services, semantic web, intelligent agents, discovering, composition, planning, model checking.

Abstract: This paper reviews the existing techniques used in the discovering and composing of services. The task of selecting an adequate service can quickly grow tedious if all services that are listed under a certain description have to be compared manually for the final selection. And what is more, the final selection does not only depend on service parameters like executions costs or accuracy, but depends on the usefulness of objects or information that service offers. This problem is present in open environments where entities like web services or agents need to locate other entities to achieve cooperation, delegation or interoperation. For these reason these two approaches, web services an agents have deal with these problem proposing an automated and efficient mechanism to determine a structural and semantic match descriptions between entities.

1 INTRODUCTION

The application of the semantic web in the area of web services has as aim a more intelligent web in which is possible to achieve a more effective communication among computers. It concentrates its efforts towards web service semantic descriptions search. Descriptive languages are not enough to describe complex relationships between ontologies. For that reason, other languages as OWL-S or WSMO have been proposed with the aim that other machine can read these descriptions and reason about how interact with the services.

Automatic location of services can considerably reduce the cost of making applications that work together and enable a more flexible integration, where providers are dynamically selected based on what they provide and other non-functional properties. To deal with these issues agent orientation is an appropriate design paradigm to enforce automatic and dynamic collaborations, especially in e-business and complex environments.

In this paper we present a revision of the different solutions that have contributed to deal with problem of the service discovery. The rest of the paper is structured as follows. Section 2 gives a service discovery description and it also presents the fundamental stages that are involve in the discovery process. Section 3 introduces the main approaches used in matchmaking algorithms. Section 4 surveys the existing state of the art in web service matchmaking algorithms. Section 5 the contribution of agent systems to service discovery process and we close with a summary and concluding remarks in section 6.

2 SERVICE DISCOVERY

Semantic service discovery consist of searching service descriptions with exact or similar properties. These properties, in most cases, are IOPE's (Inputs, Outputs, Preconditions and Effects). Basically, the matchmaking process consist of bounding the number of possible matches between offered and requested services. This matchmaking process can be divided in three stages:

Selection Process. A service request is received and sent to a matchmaker. The matchmaker is responsible of finding a suitable service or set of services according with the requested description. The suitability de-

pends on the information that the algorithm considers. Usually, the degree of similarity between service description and the request is used and it will depend on the degree of similarity between input and output parameters (IO' s).

Ranking. The set of possible service providers is refined according to additional information that the client has defined previously to choose the more suitable provider. The concept of non functional attributes, in most of cases related to quality of service (QoS), is introduced. These attributes are used in ranking functions that gives a mark or punctuation to each provider selected in the previous stage. Furthermore, the user can define an threshold to filter the obtained services.

Evaluation. When the results of the matchmaking process are obtained, it is recommended evaluate these results to identify possible modifications in some matchmaking parameters. The most important points to be evaluated are the quality of the results and the system execution. Precision and recall are the most commonly used measures.

3 SERVICE DISCOVERY APPROACHES

At the beginning, all the approaches used syntactic similarity to establish the degree of matching between two service descriptions. With the arrival of semantic web, service descriptions include data structures and also relationships between other concepts, restrictions and rules. At this situation, the proposed algorithms are based on semantics. Furthermore, in some complex environments, where negotiations and protocols to interact between entities, it can be possible to establish discovery models based on them.

Service Discovery based on keywords. Services can be filtered by doing a search based on keywords. A typical keyword scenario is a search engine that receives keywords as a query and the engine engages query keywords with service description keywords. In (Bachlechner et al., 2006) there is a review of this kind of discovery process.

Service Discovery based on semantics. Kywords do not use explicit semantic and, therefore, they do not allow to make inferences to achieve better search results. The use of vocabularies with a formal and explicit semantic is considered as a second approach. Ontologies give us a shared and explicit terminology to describe web services and queries with a logic formalization that will allow the use of inference. We found different approaches according to the service description language used. Among them, the most important ones use OWL-S, WSMO and SAWSDL (based on WSDL-S) languages. These languages provide the answer to the main questions that arise when a web service has to be described: What are the service requirements from the users? What the service provides to the users? How does the service work? How the service can be used?.

Service Discovery based on complex discovery model and negotiation/contracting. It comprises discovery models based on interaction protocols, forwarding QoS and privacy requirements, negotiation dialogs for refining discovery and establishing service requirements. This kind of service discovery is also being used in multiagents systems (Caceres et al., 2006) and in e-market environments.

4 SERVICE DISCOVERY BASED ON SEMANTICS

There are many matchmaking proposals in the area of web services and semantic web. In this section, we review some of them paying attention to characteristics such as the information that they use to deal with the matchmaking process, if they take into account service composition or cross ontologies or if they use QoS information during the discovery process.

4.1 IOPE's Algorithms

In general, most of the existing matchmaking algorithms use just inputs and outputs to determine the matching degree between requests and advertisements. But there are some algorithms that, apart from that information, take into account preconditions and postconditions (IOPE parameters).

The first service discovery algorithm is based on DAML-S (OWL-S predecessor) and uses IO's. It was developed by Paolucci et al. (Paolucci, 2002). This approach uses the semantic of the Service Profile and UDDI registries to maintain the descriptions of the services. The algorithm deals with the importance in matchmaking classification of the service outputs. A matching between a service advertisement and a service request consists of matching all the service request outputs with those of the service advertisement; and all the inputs of the service advertisement

with those of the service request. The degree of similarity between service provider and server request will depend on the degree of similarity between input and output parameters (IO' s) and it is reduced generally to the minimal distance between them in the taxonomic tree. The denomination of the degrees varies according to literature (Abela and Montebello, 2002)(Lei and Horrocks, 2003)(Paolucci, 2002)(Constantinescu and Faltings, 2002). Paolucci's algorithm is limited to discover simple services. It does not consider the composition discovery nor the use of different ontologies. Another lack is that the process of matching does not consider parameters related with quality of service (QoS). From this algorithm arose others that made some modification to deal with some of its deficiencies (Abela and Montebello, 2002)(Klusch et al., 2006)(Aversano et al., 2004)(Cardoso and Sheth, 2002).

(Wolf-Tilo and Matthias, 2003) define a different matchmaking algorithm. In this algorithm, they make first a search based on keywords and, once obtained a list of results, the user could introduce IO's parameters that must have the services and the values for these parameters. Finally, a reasoner eliminates those services that do not have these defined parameters and the resultants will be executed with the values that were introduced by the client. The results of the executions are ordered following some constraints that benefit the client, as the quality of service.

(Klusch et al., 2006) present OWLS-MX, a hybrid matchmaking algorithm that computes the degree of semantic matchmaking for a given pair of service advertisement and request by successively applying five different filters: exact, plug-in, subsumes, subsumed-by and nearest-neighbor. The first three are logic based only whereas the last two are hybrid due to the required additional computation of syntactic similarity values. The objective of hybrid semantic web service matching is to improve semantic service retrieval performance by appropriately exploiting means of both crisp logic based and approximate semantic matching.

In the matchmaking process is also important to consider the global schema of execution, which is given by the choreography. The task of selecting a web service, that should play a role in a choreography, implies verifying two things: the conformance of the service to the specification of a role of interest (guarantees that the message exchange will produce correct and accepted conversations), and that the use of that service (allows the achievement of the goal).

In (Baldoni et al., 2007) is shown that performing a match operation by operation does not preserve the global goal. They also show how to overcome these limits by exploiting the choreography definition. Actually, it is possible to extract from the choreography some information that can be used to bias the matching process so that the global goal will be preserved.

4.2 Composition

The algorithms above presented address the matchmaking process. However, they are limited to discover a single service. In many situations, queries that cannot be satisfied by a single service might be frequently satisfied by composing several services.

(Aversano et al., 2004) display a discovery algorithm that analyses DAML-S service profile, takes as objective the outputs of the user request and considers the possibility of reaching it with only one service. If it is not possible, the method includes a backward-chaining algorithm with the purpose of verifying the possibility of finding a match for the user request by means of a composition of several services. This algorithm is also capable of performing a cross ontology matching for service descriptions that use different ontologies. However, it crosses ontologies at query time, hence severely affecting the efficiency of the whole procedure.

The matching algorithms described until now are based on DAML-S/OWL-S and use the service profile. Analyzing web services only through their service profile (i.e., their IOs), can severely affects the process of discovery of service aggregations that satisfy a request. Indeed, the service profile does not describe the internal behavior of services, so in some cases it does not provide valuable information needed for composing services.

The first discovery algorithm based on the analysis of the OWL-S Process Model was proposed by (Bansal and Vidal, 2003). It stores advertisements of services as tree structures corresponding to their process models. The compound processes correspond with intermediate nodes whereas the atomic processes correspond with the leaves. The matchmaking algorithm begins in the root of the tree of the advertisement of the service and visits all the subtrees finishing in the leaves. For each node, the algorithm verifies the compatibility between the IOs and the IOs of the request.

SAM (Brogi et al., 2003) is an extension of the Bansal algorithm that return, when a complete matchmaking is not possible, a list of partial matchings (a composition of subservices that can provide only certain requested outputs by the client). Besides, when it does not find any match, SAM is able to suggest to the user additional inputs that can be enough to reach complete match. The main lack of this algorithm is

that does not consider the use of different ontologies.

Another interesting point to take into account is related with goals. Languages as WSMO, that consider goals to achieve a composition. The work presented in (van Riemsdijk and Wirsing, 2007) points out how goal-oriented techniques, which increase flexibility in handling failures, can be applied in the context of service-oriented systems and specifically in web services composition.

4.3 Crossing ontologies

Currently, individual users or user communities hope to be able of making queries about interesting services using descriptions that are expressed in terms of their own ontologies, which do not have to fit in with the searched service descriptions. The abovenamed algorithms do not address properly the problem of crossing ontologies.

(Cardoso and Sheth, 2002) propose an algorithm that allows to manage multiple ontologies. The similarity function used to compare concepts is based on the ontology taxonomy. The algorithm tries to manage concepts that are not related using the concepts properties. (Aversano et al., 2004) present a cross ontology matching that can cancel the problem of different ontologies. Therefore, there is no need of classify in a semantic domain the web service when creating the service description.

(Pathak et al., 2005) propose an ontology mapping during service discovery, such that terms and concepts in the service requester's ontologies are brought into correspondence with the service provider's ontologies. To do the mappings they use interoperation constraints, i.e. a set of relationships that exist between elements from two different hierarchies.

(Brogi et al., 2006) present an extension of SAM based on hypergraphs who allows to cross different ontologies. The matchmaking system consists of two main modules: the Hypergraph Builder and the Query Solver. The Hypergraph Builder analyzes the ontology-based descriptions of the registry-published services in order to build a labeled directed hypergraph, which synthesizes all the data dependencies of the advertised services. The vertexes of the hypergraph correspond to the concepts defined in the ontologies employed by the analyzed service descriptions, while the hyperedges represent relationships among such concepts (subConceptOf, equivalentConceptOf and intra-service dependency). The Query Solver explores the hypergraph by suitably considering the intraservice and inter-service dependencies to address the discovery of (compositions of) services as well as by considering the subConceptOf and equivalentConceptOf relationships to cope with different ontologies.

4.4 Hypergraphs

A hypergraph is a generalization of a graph, where edges (hyperedges) can connect any number of vertexes. Formally, a hypergraph is a pair (V,E) where V is a set of nodes or vertexes and E is a set of nonempty subsets of V called hyperedges. While graph edges are pairs of nodes, hyperedges joints arbitrary sets of nodes.

(Yang et al., 2005) use arc-labeled and arc-weighted trees to represent product/service requirements and offers. They propose a tree similarity algorithm that traverses input trees top-down and then computes their similarity bottom-up. During tree similarity computation, when a subtree in T_1 is missing in tree T_2 (or viceversa), the algorithm compute the simplicity of the missing subtree. The tree simplicity measure takes into account the node degree at each level, the depth of the leaf node and the arc weights. This algorithm allows partial product descriptions representations via subtrees missing.

(Hashemian and Mavaddat, 2005) use a specific notation, called interface automata (state-base model) in order to formally model web services. The information that an interface automaton exposes is the IO of a component and the temporal ordering of the actions it performs. This information can be extracted form the OWL-S specification of web services. Based on the properties exposed by interface automaton of each web service ws, three pieces of information are stored in the repository: its set of inputs, its set of outputs and dependency information between IOs of the web service. The repository is stored as a graph that contains web services information. The nodes represents I/O and there is a directed edge form node v_1 to node v_2 if and only if there is a dependency between the input and the output. They solve the problem in two steps: (i) finding web services that can potentially participate in the composition, and (ii) finding the composition setup based on the web services found in the previous step.

4.5 Model checking

Web services are composed online from pieces of software created by different programmers. Individual services can be checked to ensure that they are error free, but when new services are composed there are no means to check whether the composed service fulfils its purpose. Some formal methods, as model checking, has been proposed to verify the correctness

of complex services. But current languages are semi-formal, so the correctness of the composition depends on the cleverness of the designer. To use formal models requires translations from the languages used to describe WS into more formal ones.

(Gao et al., 2006) translate web services specified in BPEL4WS into pi-calculus, which is nearer to programming languages than finite automata or temporal logics. Nevertheless, this formal description is not soundness and some manual translation is still needed. The model checking is used with two purposes: (i) to check if services satisfy customer's demands and and designer's specifications, and (ii) to check if orchestration satisfies liveness, safety, fairness and reachability. Different methods are used: bisimulation to verify the specification, mu-calculus to check properties as safety or reachability and pi-calculus to eliminate ill behaviors.

Nakajima claims that to verify a composite web service prior to its execution may be mandatory (Nakajima, 2002). First, translates a WSFL description into *Promela*, the specification language for SPIN model checker. Furthermore, additional properties are expressed in LTL to be added to the model checking process. The verification process detects reachability, deadlock freedom and specific user properties.

Planning as model checking (Giunchiglia and Traverso, 1999) is a method of solving planning problems modeling them as model checking problems. This solution is based on transition systems, but web services are a message passing paradigm, so some special considerations have to be made. (Yu and Reiff-Marganiec, 2006) make a formulation of the solution by modifying the strong cycle planning algorithm, which guarantees that all paths reach a solution and they are fair. A four-phased algorithm is proposed. First, the planning goal and the initial knowledge is specified. After that, automatically selects from the repository relevant web services to build the plan. In third place, the algorithm search for plans. Finally, a physical composition step allows clients to choose the better plan, generates a executable plan specified in BPEL and monitors its execution, re-planning when a failure is detected.

(Walton, 2004) uses model checking to validate the correctness of communication protocols between agents in an platform that integrates agents and web services. The services are described in WSDL. Complex interactions among the entities that offer services are represented by the protocols, who are specified in a directly executable language called MAP. As the Nakajima's algorithm, the specification is translated into *Promela* language. This one provides a complete automatic translation that allows non-expert to validate their services.

The main problem in all these approaches is the complexity of the state space. All of them make different simplifications to the problem to be capable of managing the validation process by limiting the number of services (or agents) and the length of the message interchanging mainly. Moreover, designer's intervention is often needed to translate service descriptions into formal languages for model checking.

4.6 Non-functional parameters

In some algorithms, the service selection process is based in non-functional parameters and, in other cases, the algorithms refine the set of candidate service providers based on user-specified non-functional attributes, namely Quality of Service (QoS). These factors and domain specific characteristics affect on the service selection.

According to (J.Radatz and Sloman, 1988) the Quality of Service is a set of non-functional attributes that may impact the service quality offered by a web service. The main problem of this kind of information is that we cannot trust the QoS characteristics published by provider.

(Pathak et al., 2005) establish a categorization in two groups: domain-dependent and domain-independent attributes. The domain-independent attributes represent those QoS services characteristics which are not specific to any particular service (for instance, scalability or availability). On the other hand, domain-dependent attributes capture those QoS properties which are specific to a particular domain and most of the times are dynamic and depends on the instant in which the service is executed.

To deal with information reliability, service reputation is an specially interesting property that could be regarded as a measure that accumulates a user opinion about QoS in general. (Kalepu et al., 2004) address this problem.

(Pathak et al., 2005) propose a taxonomy for the non-functional attributes which provide a better model for capturing various domain-dependent and domain-independent QoS attributes of the services. These attributes allow users to dynamically select services based on their non-functional aspects. This work also introduce the notion of personalized ranking criteria, which enhances the traditional ranking approach, primarily based on the degree of match. Furthermore, in this work a kind of ontology mapping is presented .

Following this trend, (Kokash, 2005) present an approach based on the application of a distributed recommendation system to provide QoS information and

on testing of retrieval methods on service specifications.

(Aversano et al., 2004) base the searching process on syntactic information and on service quality metrics and semantics to increase the precision of the discovery process. To take into consideration the QoS in the discovery process, the customer of the service will assign a weight to every attribute. For each selected service, a weighted sum is performed among all the attributes an the final value represents the quality of the service.

4.7 Efficient Matching

Due to the complexity of the underlying semantic reasoning, matching semantic web service capabilities is a heavy process. Furthermore, the matchmaking process could be intractable when the number of available services gets large. (Mokhtar et al., 2006) describe a solution towards the efficient matching of semantic service capabilities. This approach combines optimizations of the discovery process at reasoning and matching levels. Towards the optimization of the discovery process at reasoning level, they use the solution proposed by (Constantinescu and Faltings, 2002) for encoding concept hierarchies, represented by hierarchies of concepts, using intervals. These hierarchies represent the subsumption relationships between all the concepts in the ontologies used in the directory. The main idea is that any concept is associated with an interval. Under the assumption that service advertisements and service requests already contain the codes corresponding to the concepts that they involve, semantic service reasoning reduces to a numeric comparison of codes.

(Srinivasan et al., 2004) present an approach to optimize service discovery in a UDDI registry, augmented with OWL-S for the description of semantic web services. This approach is based on the fact that the publishing phase is not a time critical task. Therefore, the authors propose to exploit this phase to pre-compute and store information about the incoming services. This proposal increases the time spent for publishing service advertisements, but it considerably reduces the time spent to answer a user request compared to approaches based on on-line reasoning.

The majority of the algorithms presented in the area of web services faces the discovery problem using the IO's parameters and trying to solve problems such as service composition or cross ontologies and also in some of them non-functional parameters are taken into consideration. But they do not explote all the possibilities that semantic offers.

Alg.	Lang.	IO'S/ IOPE'S	C.	Cross Ont.	QoS
Paolucci02	DAML-S	IO's			
Abela02	DAML-S	IO's			
Constant.02	DAML-S	IO's	√		
Cardoso02	DAML-S	IO's		√	√
Lei03	DAML-S	IO's			
Wolf03		IO's			Soft constr.
Bansal03	DAML-S	IO's	√		
Brogi03	OWL-S	IO's	√		
Benata.03	DAML-S	IO's	√		
Aversa.04	DAML-S	IO's PE's?	√	√	√
Sriniv.04	OWL-S	IO's		√	
KluS05	OWL-S	IO's PE's?	√		
Pathak05	OWL-S	IOPE's		√	√
Hashem.05	OWL-S	IO's	√		
Yang05	OWL-S	IO's	√		
Kokash05	WSDL	IO's			√
Klusch06	OWL-S	IO's		√	
Brogi06	OWL-S	IO's	√	√	
Mokhtar06	DAML-S	IO's	√		

Table 1: Service Discovery Algorithms

5 AGENTS AND DISCOVERY PROCESS

Nowadays, service oriented computing (SOC) brings additional considerations, such as the necessity of modeling autonomous and heterogeneous components in uncertain and dynamic environment. Such components must be autonomously reactive and proactive yet able to interact flexibly with other components and environments. Agent orientation is an appropriate design paradigm to enforce automatic and dynamic collaborations, especially for systems with complex and distributed transactions. Agent paradigm has technical advantages in software construction, legacy systems integration, transaction-oriented composition and semantics-based interaction. For this reason, many ideas from the research in multiagent systems could be used in service-oriented computing approaches. Key MAS concepts are reflected directly in SOC with ontologies, process models, choreography, directories and facilitators, service level agreements and quality of service measures.

Agent paradigm has presented several ideas that can be taking into consideration for web service discovery algorithms. (Caceres et al., 2006), presents an approach that complement the existing methods by considering the types of interactions and roles that services can be used in. Other approaches (Sensoy et al., 2007) use the objective experience data of

agents in order to evaluate its expectation from a service provider and makes decisions using its own criteria and mental state. Basically, service consumers collect previous experiences form other service consumers with similar demand and make decisions using different methods. By simply sharing their experiences, service consumers lead to the emergence of a consumer society in which increase the overall satisfaction.

In some MAS, brokers are used as the mechanism of discovery and synchronization mechanisms among autonomous agents. (Sycara et al., 2004) present a brokering protocol which consist on two complex reasoning tasks: discovery and mediation. Discovery task is divided in two different reasoning tasks: abstraction from the query to the provider's required capabilities and comparison and matching of required capabilities with providers capabilities. Mediation task requires the broker to transform the original query into one that can send to the provider.

Sometimes, matchmaking process is considered as a query to a static set of available options. This view is to simple. We can consider matching as a typical example of symmetry and iterative nature (Heep, 2006). Symmetric because the visibility of characteristics might equally depend on whether the other party meets specific criteria. Iterative because we learn the option space by analyzing our initial query's result set, and might restrict or weaken our requirements and preferences in response. Negotiation protocols are another mechanism used in MAS. The participant agents negotiate about the properties of the services they request and provide to bind agreements and contracts with each other. In (Dang and Hungs, 2006) is presented a protocol that supports many-to-many negotiation in which several agents negotiating with many other agents simultaneously using colored petri nets.

6 CONCLUSION

After the review process, some common weakness have been detected. Most of them they do not exploit all the data provided in the service profile. The majority of the algorithms use IO's but forget the preconditions and postconditions and the capability to make inference about this data to obtain new information that could be useful to eliminate false suitable services. Moreover, only a few take into account information related with QoS and theyuse this information just for ranking, they are not used to select services or reduce the search space with this data. Furthermore, they not emphasize in other aspects that could give them more flexibility in the discovery process such as allowing partial matching or fuzzy data. In general, web services nor have internal state and neither awareness of changes in its environment and their algorithms do not consider temporal constraints over services.

Current web services technologies have not exploited sufficient semantics and approaches to dynamic service-oriented operations in open environments. Agents present an extended proposals that provide a more flexible and efficient matchmaking process. Some of the proposals use algorithms very similar to web services but they add use the characteristics of the agents to achieve a more efficient and flexible matching. We have seen in this article that the use of roles, interaction patterns, trust or negotiation. Agents solve problems present in web services such as internal state and communication. Agents have awareness of theirs internal states and for communication use asynchronous message interchange which allows to stablish conversations. There are some open issues such as consider QoS in matchmaking process and, as in web services, take into account temporal constraints.

To conclude, after analyse the proposals from these two worlds, the highest benefit would be achieved combining the advantages of each world. The problem of service discovery is faced by the web services using semantic languages, taking into account service composition and other points as crossing ontologies or QoS. But these items provide a functional matching that do not consider other information, more subjective or in a higher level of abstraction, but that can be also important to provide a more accurate service discovery that fits better with the user request. To achieve that, we have to consider no only the information present in the current descriptions of web services. We have also to consider the kind of interactions between the entities, the type of role that is necessary to interact with or the reputation of the involved providers. These characteristics are useful to limit the possible candidates to be a suitable provider and to make easier to find what you want. In some web services proposals, these ideas are being taken into account, but it is difficult task due to the difference between web services and agents, although gradually the difference between them is becoming lower.

7 Acknowledgements

This research has been supported by TIN2006-14630-C03-01 and CONSOLIDER-INGENIO 2010 under grant CSD2007-00022

REFERENCES

Abela, C. and Montebello, M. (2002). Daml enabled web services and agents in the semantic web. In *WS–RSD'02*, Germany.

Aversano, L., Canfora, G., and Ciampi, A. (2004). An algorithm for web service discovery through their composition. In *IEEE International Conference on Web Services (ICWS04)*.

Bachlechner, D., Siorpaes, K., Lausen, H., and Fensel, D. (2006). *Web Service Discovery - A Reality Check*.

Baldoni, Baroglio, Martelli, Patti, and Schifanella (2007). Service selection by choreography-driven matching. In *ECOWS Workshop on Emerging Web Services Technology*.

Bansal, S. and Vidal, J. (2003). Matchmaking of web services based on the daml-s service model. In *Second International Joint Conference on Autonomous Agents (AAMAS03)*. T. Sandholm and M. Yokoo.

Brogi, A., Corfini, S., Aldana, J., and Navas, I. (2006). Automated discovery of compositions of services described with separate ontologies.

Brogi, A., Corfini, S., and Popescu, R. (2003). Composition-oriented service discovery. *Proceedings of 5th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

Caceres, C., Fernndez, A., Ossowski, S., and Vasirani, M. (2006). Role-based service description and discovery. In *International Joint Conference on Autonomous Agents and Multi-Agent Systems*.

Cardoso, J. and Sheth, A. (2002). Semantic e-workflow composition. *Journal of Intelligent Information Systems (JIIS)*.

Constantinescu, I. and Faltings, B. (2002). Efficient matchmaking and directory services. Technical Report No IC/2002/77.

Dang, J. and Hungs, M. (2006). Concurrent multiple-issue negotiation for internet-based services. In *IEEE Internet Computing*, number Vol.10 - 6, pages 42–49.

Gao, C., Liu, R., Song, Y., and Chen, H. (2006). A model checking tool embedded into services composition environment. In *GCC '06: Proceedings of the Fifth International Conference on Grid and Cooperative Computing (GCC'06)*, pages 355–362, Washington, DC, USA. IEEE Computer Society.

Giunchiglia, F. and Traverso, P. (1999). Planning as model checking. In *ECP*, pages 1–20.

Hashemian, S. and Mavaddat, F. (2005). A graph-based approach to web services composition. *In IEEE Computer Society*.

Heep, M. (2006). Semantic web and semantic web services. In *IEEE Internet Computing*, number Vol.10 - 2, pages 85– 88.

J.Radatz and Sloman, M. (1988). A standard dictionary for computer terminology: Proyect 610. *IEEE Computer*.

Kalepu, S., Krishnaswamy, S., and Loke, S. (2004). Reputation = f(user ranking, compliance, verity). In *Proceedings of the IEEE International Conference on Web Services*.

Klusch, M., Fries, B., and Sycara, K. (2006). Automated semantic web service discovery with owls-mx. In *Proceedings of 5th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Hakodate, Japan.

Kokash, N. (2005). Web service discovery with implicit qos filtering. In *Proceedings of the IBM PhD Student Symposium*.

Lei, L. and Horrocks, I. (2003). A software framework for matchmaking based on semantic web technology. In *Twelfth International World Wide Conference (WWW2003)*, Germany.

Mokhtar, S., Kaul, A., Georgantas, N., and Issarny, V. (2006). Towards efficient matching of semantic web service capabilities. In *Proc. of WS-MaTe06*.

Nakajima, S. (2002). Model-checking verification for reliable web service. In *OOPSLA 2002 Workshop on Object-Oriented Web Services*, Seattle, Washington.

Paolucci, M. (2002). Semantic matching of web services capabilities. In *The First International Semantic Web Conference*.

Pathak, J., Koul, N., Caragea, D., and Honavar, V. (2005). A framework for semantic web service discovery. In *WIDM'05*, Germany.

Sensoy, M., Pembe, C., Zirtiloglu, H., Yolum, P., and Bener, A. (2007). Experience-based service provider selection in agent-mediated e-comerce. In *Engineering Applications of Artificial Intelligence*, number 3, pages 325–335.

Srinivasan, N., Paolucci, M., and Sycara, K. (2004). Adding owl-s to uddi implementation and throughput. In *In Workshop on Semantic Web Service and Web Process Composition*.

Sycara, K., Paolucci, M., Soudry, J., and Srinivasan, N. (2004). Dynamic discovery and coordination of agent-based semantic web services. In *IEEE Internet Computing*, number Vol.8 - 3, pages 66–73.

van Riemsdijk, M. B. and Wirsing, M. (2007). Goal-oriented and procedural service orchestration - a formal comparison. In *In MALLOW-AWESOME'007*.

Walton, C. (2004). Model checking multi-agent web services. In *Proceedings of the 2004 Spring Symposium on Semantic Web Services, Stanford, CA, USA*.

Wolf-Tilo, B. and Matthias, W. (2003). Towards personalized selection of web services. In *The Twelfth International WWW Conference en Budapest*.

Yang, L., Sarker, B. K., Bhavsar, V. C., and Boley, H. (2005). A weighted-tree simplicity algorithm for similarity matching of partial product descriptions. In *In Proceedings of ISCA 14th International Conference onIntelligent and Adaptive Systemsand Software Engineering*, Toronto.

Yu, H. Q. and Reiff-Marganiec, S. (2006). Semantic web services composition via planning as model checking. Tech. Report CS-06-003, University of Leicester.