Ranked Matching for Service Descriptions using OWL-S

Michael C. Jaeger[1], Gregor Rojec-Goldmann[1], Christoph Liebetruth[1]
Gero Mühl[1], and Kurt Geihs[2]

[1] TU Berlin, Institute of Telekommunication Systems,
FR6-10, Franklinstasse 28/29, D-10587 Berlin
{mcj, gr, trutie}@cs.tu-berlin.de
gmuehl@ivs.tu-berlin.de
[2] Univ. Kassel, FB 16, Wilhelmshöher Allee 73, D-34121 Kassel
geihs@uni-kassel.de

Abstract. Semantic Web services envision the automated discovery and selection of Web services. This can be realised by adding semantic information to advertised services and service requirements. The discovery and selection process finds matches between requirements and advertisements according to their semantic description. Based on the Web Ontology Language (OWL) an ontology for Web services (OWL-S) was introduced to standardise their semantic description. There are already some approaches available for matching of service requirements with service advertisements according to such an ontology.
We propose an algorithm, which ranks the matching degree of service descriptions according to OWL-S. Different matching degrees are achieved based on the contravariance of the input and output types for requested and advertised services. Furthermore, additional elements of the service description, such as the service category, are either covered by reasoning processes or, such as quality of service constraints, by custom matching rules. Contrary to mechanisms that return only success or fail, ranked results provide criteria for the selection of a service among a large set of results. With such a discovery mechanism additional Web services can be found that might have normally been ignored.

1 Introduction

The Semantic Web working group of the W3C develops technologies and standards for the semantic description of the Web. The goal of these efforts is to make the Web understandable by machines and to increase the level of autonomous interoperation between computer systems [9]. Several standards and languages were already introduced to address this objective. The most relevant are the *Resource Description Framework (RDF)* [5], and the *Web Ontology Language (OWL)* [11]. The primary purpose of RDF is to structure and describe existing data. Thus, RDF is also called a metadata language. RDF Schema is a vocabulary extension to RDF to describe a semantic network. Based on RDF and RDF Schema, OWL – the successor of the DARPA Agent Markup Language in conjunction with the Ontology Inference Layer, in short DAML+OIL – increases the level of expressiveness with a richer vocabulary but retaining the decidability.

These languages are primarily used to describe content. The next logical step is to describe the semantics of services in order to improve their platform- and organisation-independent interoperability over the Internet. Referring to the Semantic Web, the research field addressing this objective is named *Semantic Web services* [12]. Its vision is the application of semantic description for Web services in order to provide relevant criteria for their automated selection. Based on the predecessor of OWL – DAML+OIL – an upper ontology for the description of Web services named DAML-Services (DAML-S) had been introduced [4]. Using DAML-S, the functionality, execution structure and the binding to existing interface information can be described. Recently DAML-S has been updated and renamed to OWL-S, adapting the evolution of DAML+OIL to OWL [21].

Using OWL-S for the description of Web services can increase the ability of computer systems to find eligible services autonomously. This is important in open environments where provided services can appear and disappear dynamically. Basically a service provider describes his advertised services in an OWL-S compliant ontology and a service requester queries for services with an OWL-S ontology expressing his requirements. In this scenario, matching service descriptions of advertisements with requirements has the purpose to select a suitable service among a set of available ones. Considering known matching approaches that return either mismatch or match, this selection process has the following potential benefits:

- Consider a matching task that returns a large set of services that might meet the requirements. How can a service-seeking agent ensure to choose the optimal service? If the matching process provides ranked results instead of the conclusion that all services ”match”, the optimal service match is given by the ranking.
- Consider a matching task that returns either a match or mismatch based on a particular threshold of matching degree. If this task returns no services, the service requester might be willing to weaken his requirements in order to find at least one matching service.

Using the OWL-S ontology has particular implications that are reflected in the matching algorithm. For example, a definition about the quality rating in OWL-S could be handled using a rule based mechanism, whereas the conceptualisation of the service category is addressed by reasoning functionality. Therefore, our matching algorithm covers the following aspects:

- We consider the contravariance for the types and their subtypes of the inputs and outputs of a service. This provides another matching degree in addition to the equivalence of concepts.
- Additionally to the inputs and outputs of a service, our matching approach covers the categorisation of the service itself.
- Besides reasoning and type-matching elements of the algorithm, also custom elements are supported addressing individual constrains or requirements, such as quality of service.

The following section 2 provides an overview about the background of this research. It introduces the usage scenarios of a matchmaker and the basics of OWL-S. In section 3

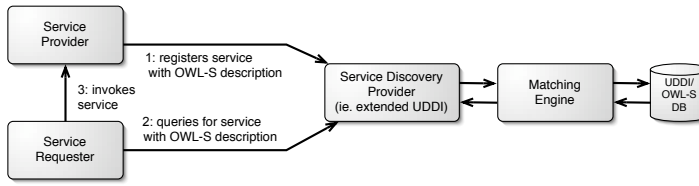

Fig. 1. Matchmaking in a Server-Sided Scenario

we present our novel matching algorithm that is based on OWL-S, and in section 4 we briefly mention some issues with the implementation of the algorithm. The section 5 discusses the algorithm and compares it to related work. At the end, in section 6, we summarize our conclusions and give an outlook to future work.

2 Background

The matchmaking of semantic descriptions is part of the discovery process of Web services. In the architecture description of Web services by the W3C [3] three main discovery scenarios are identified: a centralised registry, an index, and a peer-to-peer (P2P) scenario.

In a registry scenario, a central authority stores service descriptions, which were explicitly submitted by the service providers. An existing specification for such a repository is UDDI [19]. To take also semantic descriptions into account, existing repositories can be extended to be compatible with OWL-S descriptions. When a service requester submits his requirements, the server processes this request by matching the requirements with the description of advertised services. Then, the server returns the found match(es). This scenario is outlined in figure 1. The approach to extend UDDI repositories with semantic description such as DAML-S, is already introduced in [13] and [1].

Another approach is the index scenario. An index just provides the information to find services, whereas service descriptions are provided elsewhere. This approach can be compared with so called crawlers or robots that browse the Web automatically to create an index [8]. The difference between the index and the registry approach is that service providers control what information is put into the registry, whereas the index gathers information on its own, in most cases automatically. In [18] Sycara et. al. explain, how a broker between service requestor and provider finds matches based on OWL-S descriptions. Referring to the basic Web service architecture schemes in [3], a broker stores an index and represents a single point of contact for a service requestor, but acts as an agent to perform the matchmaking of service advertisements and requests.

In a P2P scenario each Web service is being discovered dynamically. A service requester queries other nodes in its network or a specific network domain to find and identify suitable Web services. Depending on the P2P architecture they do or do not have a centralised registry or index. A missing central repository could mean more complexity to identify matching Web services but more reliability and less organisational efforts

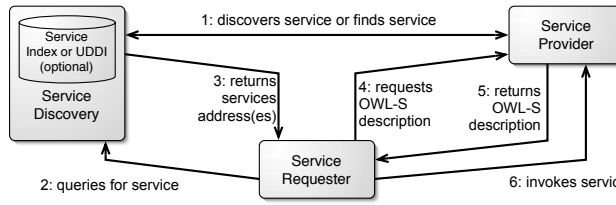

Fig. 2. Matchmaking in a Client-Sided Scenario

to find Web services. Another advantage is that in such an architecture service descriptions are always up to date, whereas a registry or an index might provide outdated information. However, P2P architectures might comprise caching or index mechanisms, or description repositories. If these components are distributed over the P2P network, the validity of the information might be even worse than with using a centralised approach. We presume, that if Web services are discovered in an index scenario or on a P2P basis, the service requester obtains the OWL-S description for eligible services and performs the matching process on its own. Figure 2 outlines a scenario, where the client processes the OWL-S descriptions.

Performing the matchmaking process either on a centralised server or by individual clients has different characteristics. As an advantage of the server-sided scenario, the implementation of a client, which acts as the service requester can be kept very simple. Thus, the effort for finding services on the client side is very low. This issue is a well known criterion for a server-oriented architecture in general.

However, a criterion to prefer the client-sided approach is the following scenario: A service requester wants to use an own matching algorithm instead of the implementation on a central server. These custom modules can define constraints that must be satisfied by the OWL-S description of the advertised service. Only with the client-sided execution of such modules the personalisation of the matching process is possible. Since our approach includes user-customisable modules to enhance the matching algorithm, we have decided in favour of a client-sided scenario. In our approach a service requester queries the OWL-S descriptions from an already obtained list of service descriptions to find the best match. However, our algorithm can also be used on a central repository, if all relevant client information is shipped via the repository interface.

2.1 Short Introduction to OWL-S

The OWL-S ontology defines three basic elements: (1) a service profile to describe the functionality of a service, (2) a process model to describe the structure of the service, and (3) a service grounding to map the abstract interface to concrete binding information. As the focus of this work is on the discovery and selection of Web services, we will concentrate on the service profile, which provides the necessary elements for our matching algorithm.

The profile is divided into three main sections: (a) a textual description and contact information, which is mainly intended for human users, (b) a functional description

of the service. This functional description describes the input and output of a service. Additionally, two sets of conditions are defined, namely preconditions, which have to hold before the service can be executed properly, and effects, which are conditions that hold after the successful execution of the service, i.e. postconditions. These four functional descriptions are also referred to as *IOPE* (Input, Output, Precondition, and Effects). In this version of our matching algorithm we will cover only the input and outputs, because preconditions and effects are not yet sufficiently standardised for being considered by a matching algorithm.

The third type (c) is a set of additional properties that are used to describe the features of the service. From these properties we use the service category, which is used to classify the service with respect to some ontology or taxonomy of services. On an optional basis, other properties can also be taken into account, such as the element QualityRating, or custom defined properties, such as the duration of the execution.

3 Ranked Matching for OWL-S Descriptions

The OWL-S specification defines the semantic elements for advertising the functional description of a service with an instance of the class Profile. Although a service can have multiple profiles as a Web service can provide multiple operations, in the following the words service and profile are used synonymously, meaning one specific profile definition. In the class Profile RDF properties point to the IOPE elements Input, Output[1], Precodition, and Effect. Each of the four classes is a subclass of the class Parameter. For the categorisation of their type, each instance is specified with the property parameterType. This property can point to elements of an arbitrary conceptualisation. The OWL-S definition leaves open, whether the definition uses OWL or other conceptualisation languages. A matching algorithm can only support conceptualisations that are covered by a reasoning functionality. Therefore we assume that the conceptualisation is provided in OWL, so that an OWL reasoner can determine a subsumption relation or the equivalence of concepts.

Apart from the IOPEs, a Web service can be described by additional elements like the service category. As found in the definition of the IOPE types, it is also left open, whether the definition for service categories uses OWL or other conceptualisation definitions. So for this case a matching algorithm can only support conceptualisations that are covered by a reasoning functionality as well. Therefore we assume that the conceptualisation of service categories is also provided in OWL.

The conclusion of this consideration is a matching algorithm, which is divided into four stages: (a) the matching of inputs, (b) the matching of outputs, and (c) the matching of the service category. The algorithm determines the matching for each of the stages individually. The results are aggregated with a fourth stage (d), where user-defined constraints or functionality can complete the matching result. The arrangement of these elements is specified in section 3.4, while in the next sections it is explained what is matched by each element of the matching algorithm. Thus, our first task is to classify different relations between two concepts to determine the degree of matching: Let A, B

[1] In fact, OWL-S defines the class ConditionalOutput or its subclass UnConditionalOutput as the possible conceptualisations of outputs.

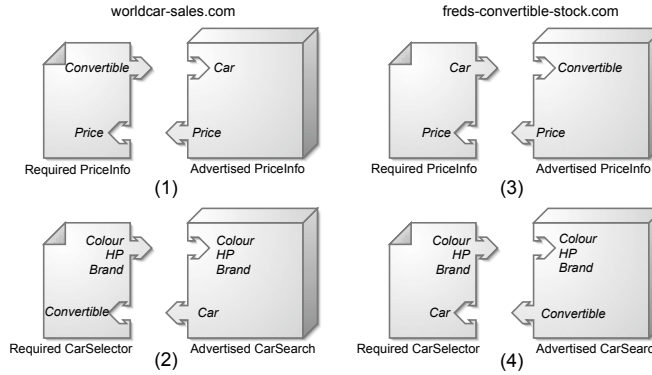

Fig. 3. Contravariance of Input and Output Types

denote two concepts which originate from a given set of ontologies. Then the following three relationships between A and B are considered:

$Fail(A, B)$	The concepts A and B are in no relation with each other.
$Subsumes(A, B)$	The concept B is subsumed by the concept A, meaning that A denotes a more general concept than B.
$Equivalent(A, B)$	A and B are equivalent, meaning that both denote exactly the same concept.

The matching algorithm performs a reasoning over the available conceptualisation to determine the relationship between the concepts.

3.1 Contravariance

For the matching of inputs and outputs, the direction of the subsumption relation is important for (a) the input types to ensure proper execution of the service and for (b) the output types to fulfill the demands of the service requester. Consider an example that is outlined in figure 3: two fictitious car selling services, ”worldcar-sales.com” (cases (1) and (2)) and ”freds-convertible-stock.com” (cases (3) and (4)), advertise two similar services. One service returns the price of a car and the other provides a search functionality to find a car by some key properties. We assume that the parameter type convertible is defined to be a subclass of the parameter type car in an available OWL ontology.

The input types of service requirements in scenarios (1) and (3) do not match with equivalent ranks to the service advertisements. In case (1) the input type of the requirement is subsumed by the input type of the advertisement, in case (3) vice versa. However, the proper execution of the service can be ensured in case (1) with the assumption, that the contravariance holds for the input type of the advertised service. For case (3) it might happen that the advertised service requires some specific characteristics for the proper execution.

Regarding the output types the subsumption direction between required and advertised service is now reversed. In case (2) the requester is particularly looking for convertibles, but might get all kinds of cars. The requirement cannot be ensured. For case (4) the requester has just looked for cars, and the service returns convertibles only. But in this case the requirements are still met assuming the contravariance for the output type of the service requirement.

3.2 Matching of Inputs, Outputs, and the Service Category

From the four stages of the matching algorithm, the first three stages match the classifications of inputs, outputs and the service category based on subsumption reasoning, because the classifications can be described as concepts of some OWL ontology. In a fourth stage custom matching modules are supported to ensure requirements that cannot be achieved with ontology-based reasoning. The ranks for each of the three first stages are summarised in table 1.

3.3 User-defined Matching

In addition to the three matching stages also a user-defined matching stage is provided. This matching stage consists of one or more additional modules, which ensure specific requirements or conditions. As a result, each of the modules can return either true, false, or a specific matching degree.

Along with the documentation of OWL-S in the Internet some examples are provided on the OWL-S homepage, which clarify the benefit of this stage. Consider the example service profile of the fictitious Airline Bravo Air named BravoAirProfile.owl found in [20]. It includes a custom parameter such as the geographic radius of a service[2] and an application of the class QualityRating, which can be used to denote a measure for quality of service statements. Applied to this example, custom matching modules can ensure thresholds for the quality rating or the availability of a service at a geographical location.

3.4 Combination of the four Stages

The result of the matching algorithm is an aggregation of the individual four stages. The simplest approach is to add the ranks of the first three stages and complete the sum with the result of the user-defined module(s). The aggregated result is either a number, which is an abstract measure of the matching quality, or the conclusion, that the descriptions did not match. This aggregation schema is outlined in figure 4.

The algorithm could be modified optionally by assigning weights to the results of the individual stages. For example, the rank of the output matching could be multiplied by two, because the result of a service invocation might be more relevant than its parameterisation. For a sound weighting of the individual stages we propose to test the matching algorithm with sets of training data.

[2] The geographic radius was part of the upper ontology in previous versions of OWL-S. For OWL-S 1.0 this element has been discarded. However – as shown in this example – the geographic radius can be defined as a custom parameter and added to the OWL-S description of a service.

#	Result	Interpretation
Inputs		
0	FAIL	At least one input type of the advertised service has not been successfully matched with one input type of the advertised service. The service cannot be executed properly.
1	UNKNOWN	This matching result is provided, if the used categorisation is not supported by the matching algorithm. However, this result is not the conclusion of an open-world-assumption approach. The reasoning process still follows the closed-world assumption, but due to the lack of understanding the conceptualisation, no conclusion can be drawn.
2	SUBSUMES	For each input type of the advertised service exactly one input type of the required service has been found, which is at least subsumed by the input type of the advertised service. This means that the advertised service might be invoked with a more specific input than expected, assuming the covariance of the input types of the advertise service.
3	EQUIVALENT	For every input type of the advertised service one equivalent input type of the required service is found.
Outputs		
0	FAIL	At least one output of the required service has not successfully been matched with an input of the advertised service.
1	UNKNOWN	This matching result is provided, if the used categorisation is not supported by the matching algorithm. As for the matching of inputs, this result is not the conclusion of an open-world-assumption approach, but due to the lack of understanding the conceptualisation, no conclusion can be drawn.
2	SUBSUMES	The output types of the required service subsume the output types of the advertised service or are equivalent to them. This means that the required service might receive a more specific output type than expected. Assuming the contravariance of the output types of the advertised service, it is still ensured, that the requirement of the service requester are met. Additionally for all output types of the required service a successfully matching counterpart of the advertised service is identified.
3	EQUIVALENT	For each output type of the required service one equivalent output type of the advertised service is found.
Service Categories		
0	FAIL	The two concepts were not successfully matched.
1	UNKNOWN	Either the description of the advertised or the required service is not classified or no reasoning functionality is available to determine matching for the types of categorisation.
2	SUBSUMES	The classification of the advertised service is subsumed by the classification of the required service. This means that the advertised service offers more specific functionality than required.
3	EQUIVALENT	The classification of the advertised service and the classification of the required service are equivalent.

Table 1. Ranking for the Matching of Inputs, Outputs, and Service Categories

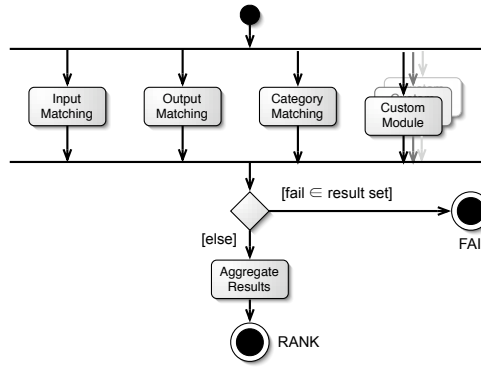

Fig. 4. Combination of the Matching Result

4 Implementation

For testing and demonstration purposes an implementation has been realised in Java, which uses the OWLJessKB reasoner from Joe Kopena[3] for performing the reasoning tasks. The work builds upon a predecessor, which was created for processing DAML-S descriptions [6][4]. The following significant changes have been applied to the existing DAML-S matchmaker to realise the new implementation:

- The handling of files, which contain the service description, has been improved. In the DAML-S version, the tool required three files, which were separated by the different parts of DAML-S *profile, process and grounding* according to its documentation [4]. Examples of this file structure can be found in the example material for DAML-S version 0.9. The new matching implementation calls only the API of the OWLJessKB reasoner component and queries all needed information in the provided knowledge base regardless of the file structure.
- A change from DAML-S to OWL-S is found in referencing concepts, which classify a service parameter. While in DAML-S the concept, which classifies a parameter, is being referenced from both the service profile and the service process model, in OWL-S only the process model references this classification, which can be defined in some OWL ontology. In the service profile a parameter is only pointing to its according description in the process model. Thus, a matchmaker for OWL-S descriptions must also process the service process model and not only the profile as a matchmaker for DAML-S. Additionally, the DAML-S matcher has used the simple subject query methods, which are provided by the DAMLJessKB reasoner API, which have been deprecated in the OWLJessKB version. For these two rea-

[3] Availble at http://edge.cs.drexel.edu/assemblies/software/owljesskb/

[4] The DAML-S version is available at http://ivs.tu-berlin.de/Projekte/damlsmatcher/, the OWL-S matchmaker at http://ivs.tu-ber.in.de/Projekte/owlsmatcher/. The software is licensed under the LGPL.

sons all queries have been reformulated to conform to the new API of the reasoner component.

The software can be used to select two sets of OWL-S descriptions, one representing a service requirement and another for a service advertisement. In a second step the user can determine the matching level, which is the required minimum for a successful match. Then, the implementation performs the matching process as introduced in the previous section. Currently, the implementation is embedded in a stand-alone Java-based GUI tool, however, the functionality could be also integrated in a service selection environment as introduced in section 2.

5 Discussion and Related Work

Our proposal is a progression from matching approaches for DAML-S ontologies, which were presented in earlier research reports. Our algorithm is based on the new OWL-S standard and provides a fine-grained and tunable ranked matching of service descriptions. A variation of the matching algorithm [10] provides ranking by supporting different reasoning operations, but considers only the categorisation of a service. Our motivation for refining the matching process is that it might easily occur that two profiles will be declared as non-compatible because one (probably less important) property of the category is not matching to a property in the other profile. An optimal matching algorithm should indicate the lack of matching quality, but should not disqualify the advertised service completely.

Furthermore, OWL-S specifications can be arbitrarily complex. By splitting up the profile and determining the matches individually for its subparts, the algorithm can identify matches between service requests and advertisements with a higher precision. The disadvantage of ignoring parts of the profile can be compensated with a proper usage of user-defined modules. For example, a module that evaluates QoS properties could easily be added to the matching algorithm. Another characteristic of our work is the consideration of the direction of the inheritance hierarchy between service requests and advertisements (”which concept subsumes which one”). This issue is not addressed for example in another matching approach that uses a custom ontology and not OWL-S or DAML-S for describing services [22].

The matchmaker ATLAS [14] and the work of Paolucii et al. [13] share our decision to consider the service profile and it’s inputs and outputs for determining matches between requests and advertisements. Our matching algorithm can be seen as an extension to these approaches, because our work adds the coverage of the contravariance for the inputs and outputs of a service. A different approach for matching DAML-S descriptions is found in the work of Bansal and Vidal [2]. A matchmaker is proposed that covers only the service model of DAML-S. We did not follow this approach yet, because as described in the introduction of DAML-S, the service model is not primarily provided to express requirements for finding matches with advertisements [4]. The matching of service descriptions is also addressed in other fields, which are not related to Web service or DAML-S. For example in [16] and [15], service descriptions are used, which are based on Conceptual Graphs. An enhanced service trader – the equivalent to

a matchmaker – performs comparisons of service specifications and returns a ranking of partial matches. Contra- and covariance rules are applied. This work has been applied for a CORBA trading service: IDL descriptions are translated into a representation using conceptual graphs, which can be processed by the AI-Trader [17]. Another approach for matching capabilities of Web services, which uses a custom RDF ontology, is found in [22].

As already mentioned some approaches also propose the integration of DAML-S descriptions with the UDDI standard, which we have introduced as an application scenario for the matching algorithm ([13] and [1]). For testing our work, the OWL-S descriptions were created by using an XML-Editor. This is by far not as efficient as probably desired by potential users of OWL-S. Thus, we also plan to supplement our work with a methodology and a tool for the creation of OWL-S descriptions as introduced for DAML-S in [7]. A similar GUI tool for the description of services based on Conceptual Graphs was developed for the AI Trader in an earlier project of our research group [15]. Such a methodology supported by a tool for the creation of OWL-S-based descriptions would increase the usability of OWL-S service descriptions, would ensure, that the relevant elements are well defined, and therefore would increase the efficiency of our service matching algorithm.

6 Conclusions

For the matching of semantic Web services we have shown that specific elements of the OWL-S upper ontology can be used to provide a fine grained ranking of matching results rather than flat subsumption reasoning for the service profile description as a whole unit. This work also indicates the potential to provide even finer grained ranks with the consideration of more elements of OWL-S.

The development of the implementation revealed the need for an efficient methodology and tools to create OWL-S descriptions. As a consequence, we have started to design a methodology and a tool. This tool shall facilitate the creation of OWL-S specifications and individual matching constraints, similar to the AI Trader tool mentioned in the previous section. In general, we strongly believe that search facilities and matchmakers will be important components in future service-oriented architectures, but users of OWL-S need tools for the creation of the OWL-S descriptions almost as easy to use as with WSDL.

References

1. Rama Akkiraju, Richard Goodwin, Prashant Doshi, and Sascha Roeder. A method for semantically enhancing the service discovery capabilities of uddi. In *In Proceedings of the Workshop on Information Integration on the Web*, pages 87–92, August 2003.
2. Sharad Bansal and Jose M. Vidal. Matchmaking of web services based on the daml-s service model. In *Proceedings of AAMAS'03*. ACM Press, July 2003.
3. David Booth et al. Web services architecture. Technical report, W3C, http://www.w3.org/TR/ws-arch/, 2004.

4. Anupriya Ankolenkar et al. Daml-s: A semantic markup language for web services. In *Proceedings of 1st Semantic Web Working Symposium (SWWS' 01)*, pages 441–430, Stanford, USA, August 2001. Stanford University.
5. Frank Manola et al. RDF Primer. Technical report, W3C, http://www.w3.org/TR/rdf-primer/, 2004.
6. Michael C. Jaeger and Stefan Tang. Ranked matching for service descriptions using daml-s. In Janis Grundspenkis and Marite Kirikova, editors, *Proceedings of CAiSE'04 Workshops*, pages 217–228, Riga, Latvia, 2004. Riga Technical University.
7. Michael Klein and Birgitta Koenig-Ries. A Process and a Tool for Creating Service Descriptions based on DAML-S. In *Proceedings of 4th International Workshop Technologies for E-Services, TES 2003*, pages 143–154. Springer, September 2003.
8. Mei Kobayashi and Koichi Takeda. Information retrieval on the web. *ACM Computing Surveys*, (2):144–173, June 2000.
9. Marja-Riitta Koivunen and Eric Miller. W3c semantic web activity. In *Semantic Web Kick-Off in Finland*, pages 27–44, November 2001.
10. Lei Li and Ian Horrocks. A software framework for matchmaking based on semantic web technology. In *Proceedings of the 12th International Conference on World Wide Web (WWW2003)*, pages 331–339. ACM Press, May 2003.
11. Deborah L. McGuinness and Frank van Harmelen. Owl web ontology language overview. Technical report, W3C, http://www.w3.org/TR/owl-features/, 2004.
12. Sheila A. McIlraith and David L. Martin. Bringing semantics to web services. *IEEE Intelligent Systems*, 18:90–93, January/February 2003.
13. M. Paolucci, T. Kawamura, T. Payne, , and K. Sycara. Semantic matching of web service capabilities. In *Proceedings of 1st International Semantic Web Conference. (ISWC2002)*, pages 333–347. Springer-Verlag, Berlin, 2002.
14. Terry R. Payne, Massimo Paolucci, and Katia Sycara. Advertising and matching daml-s service descriptions. In *Position Papers for SWWS' 01*, pages 76–78, Stanford, USA, July 2001. Stanford University.
15. A. Puder, F. Gudermann S. Markwitz, and K. Geihs. AI-based Trading in Open Distributed Environments. In *Proceedings of the 5th International Conference on Open Distributed Processing (ICODP'95)*, Brisbane, Australia, February 1995. Chapman and Hall.
16. Arno Puder. *Typsysteme für die Dienstvermittlung in offenen verteilten Systemen*. PhD thesis, Computer Science Department, Johann Wolfgang Goethe University, Frankurt/M., 1997.
17. Arno Puder and Kurt Geihs. Meta-level Service Type Specifications. In *Proceedings of the IFIP/IEEE international conference on Open distributed processing and distributed platforms*, pages 74–84, Toronto, Ontario, Canada, 1997. Chapman and Hall.
18. Katia Sycara, Massimo Paolucci, Julien Soudry, and Naveen Srinivasan. Dynamic discovery and coordination of agent-based semantic web services. *IEEE Internet Computing*, 8(3):66–73, May, June 2004.
19. UDDI Spec TC. Uddi version 3.0.1. Technical report, OASIS, http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm, 2003.
20. The OWL Services Coalition. OWL-S Example Description for Bravo Air. Technical report, http://www.daml.org/services/owl-s/1.0/BravoAirProfile.owl.
21. The OWL Services Coalition. OWL-S: Semantic Markup for Web Services. Technical report, http://www.daml.org/services/, 2004.
22. David Trastour, Claudio Bartolini, and Chris Preist. A semantic web approach to service description for matchmaking of services. In *Proceedings of the 11th international conference on World Wide Web*, pages 89–98, Honolulu, USA, May 2002. ACM Press.