Partner and Service Discovery for Collaboration Establishment with Semantic Web Services

Michael Stollberg, Uwe Keller, Dieter Fensel
DERI – Digital Enterprise Research Institute
University of Innsbruck, Austria
{michael.stollberg, uwe.keller, dieter.fensel}@deri.org

Abstract

The ultimate goal of the Semantic Web is to enable automated collaboration over the Internet, based on ontologies as semantic terminology definitions and Web Services as computational facilities accessible over the Web. An essential functionality for collaboration support on the Semantic Web is detection of entities, services, and other resources that are to be used for achieving a successful collaboration. This is commonly referred to as discovery, wherefore the emerging concept of Semantic Web Services promises more effective support than conventional Web Service technologies: based on exhaustive semantic description frameworks, intelligent mechanisms are envisioned for discovery, composition, and contracting of Web Services. This paper outlines an approach for automated collaboration support using Semantic Web Services, and presents the realization of semantically driven discovery of cooperation partners and usable Web Services as a main component for collaboration establishment.

1. Introduction

The Semantic Web, envisioned as the next generation of web technology, aims at advanced information processing along with ad-hoc combination and usage of services to allow automated interaction and collaboration over the Web. Therefore, three key technologies have been identified: ontologies for semantically enhanced information exchange, Web Services for reuse and interoperability of computational functionality, and agent technology for automated execution of tasks [2].

Therein, Web Services shall enable automated and dynamic handling and execution of computational facilities. The current Web Service technology stack allows exchange of messages between Web Services (SOAP), describing the technical interface for consuming a Web Service (WSDL), and advertising a Web Services in registries (UDDI). However, these technologies do neither support explicit functional descriptions of Web Services nor ontologies for semantically enhanced information interchange definitions. Consequently, the emerging concept of Semantic Web Services aims at providing more sophisticated Web Service technologies: on basis of semantic description frameworks, intelligent mechanisms are envisioned for discovery, composition, and contracting of Web Services. Several research efforts are concerned with elaboration of Semantic Web Service technologies, mainly concentrated around OWL-S [10] and WSMO [13] as the most significant frameworks for Semantic Web Services existing at this point in time.

The idea for collaboration support on the Semantic Web is to enable interactions of Web Services. Therefore, appropriate collaboration partners as well as usable Web Services for collaboration execution need to be detected out of possibly numerous existing parties and resources. The aim of this paper is to present efficient and high-precision mechanisms for detecting appropriate collaboration partners (Partner Discovery) and usable Web Services for collaboration execution (Service Discovery). These are core components of a system for automated collaboration support that applies emerging technologies for Semantic Web Services.

The paper is structured as follows: Section 2 identifies the functional requirements for partner and service discovery within our approach for automated collaboration with Semantic Web Services; Section 3 explains the theoretic basis and approach for realizing partner and service discovery; Section 4 presents the discovery components architecture along with testing results; finally, Section 5 discusses related work and concludes the paper.

2. Collaboration with Web Services

In order to explain the distinction and functional requirements for partner and service discovery, the following outlines our approach for automated collaboration support with Semantic Web Services. We briefly replicate the conceptual model and refer to [14] for a detailed presentation of the framework.

Based on real-world collaboration models, we assume that there are numerous entities that want to solve problems in a cooperative manner. Every entity has an objective to be achieved, and facilities as the means for performing collaborative problem solving. A collaboration between entities is considered to be potentially successful if the objectives of participating entities are compatible, and if there facilities are interoperable in the sense that an interaction of these performs collaboration execution. Figure 1 shows the system building blocks for a realization of this collaboration model.

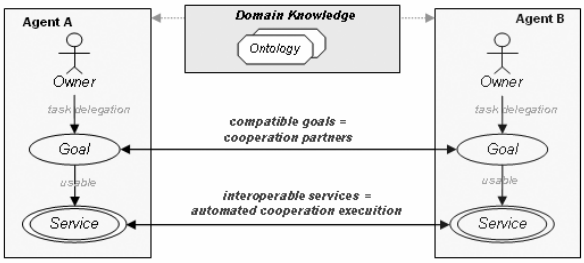

Figure 1: Collaboration Model

Agents represent entities involved with service usage and provision; an agent carries a goal that represents the objective of the entity, and Web Services that provide the facilities the entity can use for automated execution of a collaboration. All building blocks carry semantic descriptions, wherefore we apply the description notions defined within the Web Service Modeling Ontology (WSMO for Ontologies, Goals, and Web Services, [13]. An agent can be assigned with several different goals during its lifetime that are resolved separately in collaborations. According to the above definition of successful collaborations, agents are considered as potential collaboration partners if their goals are compatible and if their services are interoperable. Based on the semantic resource descriptions, different components establish and execute collaborations.

Collaboration establishment is achieved by three components shown in Figure 2. Given a society of electronic representatives with unresolved goals, the Partner Discoverer detects potential cooperation partners by determining compatibility of the Goals; concurrently, the Service Discoverer detects the services that can be used by the agent as facilities for automated collaboration execution. The results of partner and service discovery are unified into a preliminary collaboration that includes the agents as collaboration partners, their goals and a set of usable services. Then, the third component checks whether the behavior interfaces of the services detected for potential cooperation partners are compatible in order to ensure prosperous interaction of services during collaboration execution. The resulting collaboration consists of the same resources as the preliminary one, but with a reduced set of usable services; this is executed, whereby the objectives of the participating entities are resolved.

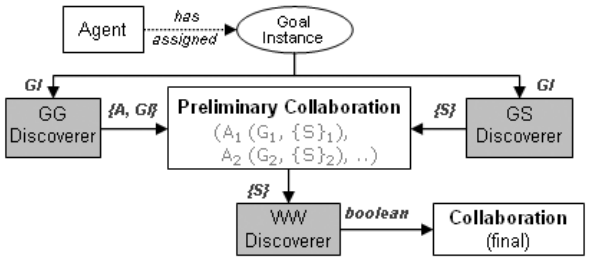

Figure 2: Collaboration Establishment

Partner and service discovery in this framework are concerned with detection of potential partners and resources of potentially numerous entities and services existing in the system. This issue, referred to as the connection problem within the agent community [3], is of significant importance for enabling cooperation establishment on the Semantic Web. The remainder of this paper presents the realization of partner and service discovery as efficient and high-precision discovery mechanisms for collaboration establishment.

3. Approach for Discovery Realization

The following explains the theoretic basis for the realization of partner and service discovery. The structure and usage of the resources applied within partner and service discovery are based on WSMO, which allows applying the theoretical framework for Web Service discovery defined within WSMO [8]. We generalize and extend this for partner and service discovery.

The prerequisite for correct and precise semantically driven discovery is provision of sufficient and well-defined knowledge. The usage of Web Service is considered as an action that results in changes of objects in the world. Hence, we distinguish two types of knowledge for resource descriptions: so-called *action knowledge* that denotes the actions performed in service execution, and *object knowledge* for describing

the items that an action is performed on. For example, in a Goal 'selling a brown chair made of wood', the action is 'selling' and 'a brown chair made of wood' the object.

Semantically enabled discovery relies on determining whether certain properties hold between resource descriptions. Therefore, different relations have to hold on the knowledge types in resource descriptions: actions need to be compatible, while object definitions need to be non-contradicting. A suitable collaboration partner for the example above would define a goal 'buying a wooden chair'; therein, 'buying' denotes an action that is compatible to 'selling', while 'a wooden chair' defines an object that is not contracting to the object of the goal described above. The following explains the means for describing resources according to the knowledge type distinction as well as the techniques applied for determining matchmaking.

3.1. Action-Resource Ontology

Action knowledge is defined in the *action-resource ontology* that specifies action items and their compatibilities as well as the relation between action items and resources. The ontology consists of a taxonomy of actions and a taxonomy of resources; an action has a set-valued attribute *compatibleAction* that references to compatible actions; a resource has a set-valued attribute *hasAction* that denotes the actions belonging to the resource type; this resource type is referenced in concrete resource descriptions, so that all resources in an application are instances of the action-resource ontology. Figure 3 gives an example of an action-resource ontology with the compatible actions 'buy' and 'sell' that are related to respective types of resources.

```
concept action
 compatibleAction ofTypeSet action
concept buy subConceptOf action
 compatibleAction ofTypeSet sell
concept sell subConceptOf action
 compatibleAction ofTypeSet buy

concept resource
 hasAction ofTypeSet action
concept goal subConceptOf resource
concept buyergoal subConceptOf goal
 hasAction ofTypeSet buy
concept service subConceptOf resource
concept buyerservice subConceptOf service
 hasAction ofTypeSet buy
```

Figure 3: Action Resource Ontology[1]

[1] The ontology is defined in WSML, the language for Semantic Web Service descriptions developed for WSMO. We refer to [4] for syntax and semantic specification of this language.

On basis of this ontology, simple inferences can be defined for determining action compatibility of resources. Although not displayed in the example, the action-resource ontology allows definitions of more complex action knowledge structures by the set-valued attributes of actions and corresponding resources, like the sale of cars of a specific brand and a specific type. The use of ontologies for semantic modeling of domain knowledge about actions and resources enables flexible matching mechanisms than predefined relations outside of such an ontology.

3.2. Set-based Object Matchmaking

Object definitions in resource descriptions rely on domain-ontologies that provide semantically unambiguous terminology of objects in the domain of discourse and relations among these [4].

As one possibility of modeling resource descriptions, we use a set-based modeling approach for logical expressions in the constituting notions of goals and service descriptions. This means that a logical expression describes a subset of the information space - that is all possible instances of the domain ontologies - that fulfills the desire or restriction intended by the description notion. For example, goal 'buy a wooden chair' specifies that all concrete instances of a domain concept chair that meet the condition of being made of wood that are applicable to satisfy the goal.

More formally, set-based modeling can be implemented in a first-order logic L as follows: A resource, e.g. a Web Service or a goal, is considered as set R of concrete instances wrt. a set of ontologies O. The set of all possible instances of the Ontologies O is called the universe U; hence, $R \subseteq U$. A (formal) *resource description* (relative to some set of Ontologies O) is a pair (D_R, r_R) where D_R is a set of closed formulae in L and r_R is a unary predicate symbol in L which satisfies the following property:

I is a model of D_R iff. $I(r_R) = R$, (1)
for every L-interpretation I which represents O.

The discovery technique applied on set-based resource models is matchmaking of object definitions in resource descriptions. Thereby, a match is achieved when the object definitions in the resources descriptions do not contradict, i.e. there is a common possible instance in U which satisfies the object definitions in all resources involved of interest. Therefore, set-theoretic criteria for object matchmaking defined in [7] have been adopted and extended to the WSMO framework into five matchmaking notions for Web Service discovery [8].

The different matchmaking notions express different kinds of matches that can be exploited for faithful object matchmaking. Each notion defines a different logical relationship between the resource descriptions considered for discovery; for example, the Exact Match defines that if the object defined in the description of a resource D_Q that is associated with a discovery request is equivalent to the object definition of a resource D_R that is associated with the discovery result, then D_Q and R_R are considered as matching. Figure 4 shows the five notions with their logical formalization and the matchmaking intentions that can be achieved with each notion. For further elaboration of the object matchmaking notions and their relationship we refer to [8].

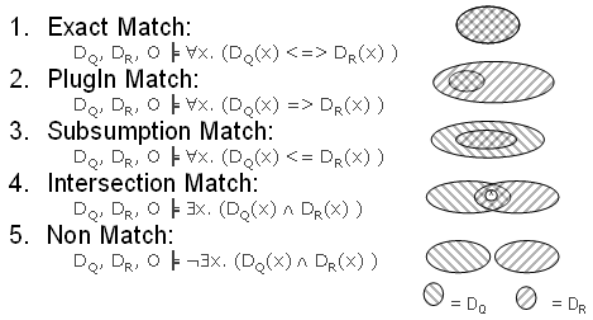

Figure 4: Set-based Matchmaking Notions

Set-based object modeling and its formal representation in a logical language *L* ensure precise and unambiguous semantics of object definitions in resource descriptions, and the applied matchmaking notions provide an accurate theoretical basis for high precision object matchmaking. Before addressing the application of the action and object knowledge discovery techniques, we first explain the technical realization of object matchmaking.

3.3. Object Matchmaking with VAMPIRE

For a research prototype, we used VAMPIRE as the technical platform for realization of object matchmaking, a resolution-based theorem prover for first-order classical logic with equality [12].

The reason for choosing a theorem prover was that the proof obligations for object matchmaking are verified by determining the provability on basis of the structure and semantics of logical formulas; the language for the description of the involved resources as well as the ontologies should not be artificially restricted for the prototype. As a consequence, the resulting proof obligations do in general not fall into a specialized subset of *L* for which alternative approach than general-purpose theorem proving can be used.

VAMPIRE has been chosen because of its excellent performance in comparison to other systems for automated theorem proving and its capabilities for ontology reasoning [16]. Since WSML has been chosen as a rich modeling language for the description of goals, Web Services and Ontologies, we implemented a translation of WSML descriptions to specifications in First-order Predicate Logic as required by the VAMPIRE system.

The only domain knowledge needed for object matchmaking with a theorem prover is the schemas of the used domain ontologies (i.e. taxonomy of concepts and relations, and axioms), and so-called *generic instances* for all ontology concepts. Roughly speaking, generic instances are formulae which represent the space of possible instances of the universe *U* relative to applied matchmaking notion. Hereby, *U* is mainly determined by the ontologies *O* which are used for the resources descriptions. Furthermore, we need the formal resource descriptions as described in Section 3.2. The proof obligations to be checked by the theorem prover for the single matching notions are given as logical entailment relations in Figure 4.

When restricting the description language for resources, it would be possible to exploit deductive database systems for checking the set-based matchmaking notions relative to a knowledge base. As a consequence, we would expect substantial performance enhancements since the reasoning task performed in deductive databases is more specialized and a lot simpler than general-purpose theorem proving. Further use cases have to show whether the required restriction of the description languages is feasible.

4. Discovery Components

After explaining the theoretical basis for semantically driven discovery, this Section presents the architecture of the components for partner and service discovery. We first outline the design principles, then explain the architecture of each component in detail, and conclude with a report on testing results.

The architecture of the discovery components follows a common structure wherein we define a Discovery Request *Q* as the input for a discoverer and a Discovery Result *R* as its output; *Q* and *R* can be associated with any type of resource that is refined in specific discoverers according to their functionality. *Q* carries all knowledge that is necessary to detect matching resources, which is comprised of the associated resource along with its reference to the action-resource ontology, and the matchmaking notion to be applied for each the object definition in the resource descrip-

tion. In consequence, *R* contains a set of resources that match *Q* with regard to the associated knowledge.

The modularized components for partner and service discovery are executed concurrently and independent of each other. Their respective discovery results are dynamically combined as a preliminary collaboration which is then further processed in the collaboration system outlined in Section 2. Each component realizes a layered architecture in order to subsequently narrow the search space and thus minimize the amount of resources that need to be considered for computationally expensive operations.

The architecture of the Partner Discoverer as well as the one of the Service Discoverer rely on the distinction of two goal notions in the system: first, so-called Goal Templates *GT* are goals predefined at design time whereof Goal Instances *GI* are created of during runtime in order to specify a concrete goal assigned to an agent. *GT*s are supposed to be reused for creating several *GI*s by refinement of a *GT*. Hence, relations between *GT*s can be pre-computed and reused across several *GI*s. Considering that in typical applications, there are only a few *GT*s while possibly numerous *GI*s are created during runtime, the components' architectures rely on the relationship of *GT*s and *GI*s in order to enhance performance.

Regarding the semantic descriptions of goals, *GT*s are described as WSMO Goals by a *postcondition* as an object definition that satisfy the objective and is expected as computational result of service usage, and an *effect* that denotes the object that is supposed to exist in the world after successful goal resolution; besides, a *GT* carries a non-functional property that references to a subconcept of goals in the action-resource ontology (see Figure 3). A *GI* inherits the description of its corresponding *GT*, whereby object definitions in the postcondition and effect can be refined in order to express to the concrete objective (meaning that either the range of attribute values can be narrowed, or concrete attribute values are defined conform to the respective ontology schema). In addition, a *GI* carries a *submission* that defines ontology instances as those objects which are intended to be submitted as input to a Web Service.

The following explains the components' architectures in more detail.[2]

[2] We have realized the components for partner and service discovery in course of the Semantic Web Fred project, wherein the collaboration system outlined in Section 2 is being developed. The components are available as open source components on the project website: http://swf.deri.at/, along with a Web Service interface for VAMPIRE at http://dev1.deri.at:8080/vampire/services.

4.1. Partner Discoverer

The Partner Discoverer detects potential collaboration partners by determining the compatibility of their goals. Within the collaboration framework, agents can be assigned with different goals during their life time, so that an agent can participate in several collaborations wherein his role is indicated by a specific goal.

According to the general structure of discovery components outlined above, the Discovery Request of the Partner Discoverer Q_{PD} is associated with a goal that has been assigned to an agent for automated resolution. The description of this goal carries a reference to a resource type defined in the action-resource ontology, and the object definitions in the respective description notions along with the matchmaking notion to be applied. The Discovery Result of the Partner Discoverer R_{PD} is a set of goals which are assigned to different agents and match Q_{PD}, thus indicating potential collaboration partners.

The structure of Goal Templates and Goal Instances together with the refinement semantics of Goal Instances implies the following relation concerning goal compatibility:

$$\texttt{If instanceOf}(GI_x, GT_x)(x = 1,2) \texttt{ then compatible}(GI_1, GI_2) \Rightarrow \texttt{compatible}(GT_1, GT_2) \quad (2)$$

The architecture of the Partner Discoverer shown in Figure 5 relies on this relation. The Goal Instance GI_i associated with the Discovery Request is instance of a Goal Template GT_i, wherefore the Cooperation Knowledge Filter detects Goal Templates $\{GT_{com}\}$ that are compatible to GI_i. Then, the GG Matcher matches GI_i with those *GI*s that are instances of a GT_{com}, detecting sets of Goal Instances $\{GI_{com}\}$ that are compatible to GI_i; each set $GI_i \cup \{GI_{com}\}$ represents a potential collaboration whereby the cardinality of the set denotes the arity of the collaboration.

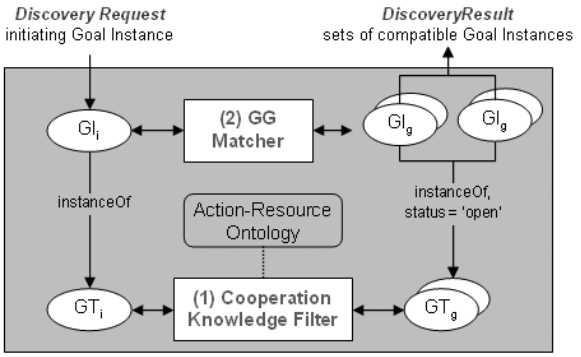

Figure 5: Partner Discoverer Architecture

As discovery techniques, the Cooperation Knowledge Filter determines action compatibility on basis of the resource type defined in the Goal Templates. The GG Matcher performs object matchmaking of the postconditions and effects of Goal Instances, applying the matchmaking notions defined in GT_i.

The Partner Discoverer supports detection of multi-party collaborations on basis of the set-valued attributes of actions and resources in the action-resource ontology. For example, GT_i is associated to the action 'purchase' and the action-resource ontology defines 'sell' and 'deliver' are defined as to be compatible with 'purchase', then goals with the actions 'sell' as well as 'deliver' are detected as a set GT_{com}; for object matchmaking, the union of the object definitions of GI_{com} has to be matched with the one of GI_i.

4.2. Service Discoverer

For collaboration establishment in our approach, the Service Discoverer detects Web Services that can be used by potential cooperation partners for automated collaboration execution. Therefore, the Service Discoverer is specified as follows: the Discovery Request Q_{SD} is associated with a goal that has been assigned to an agent for automated resolution, carrying the reference to the resource type and the object definitions along with the matchmaking notion to be applied. The Discovery Result R_{SD} is a set of services that match Q_{SD} according to the information given in request Q_{SD}.

The functional service description used for service discovery is defined in a service capability as defined in WSMO [13]. A capability is comprised of four description notions: a *precondition* defines the input requested by the service along with constraints on it, *assumptions* define constraints on the world that have to hold before the Web Service can be executed, a *postcondition* defines the output of the service with conditions on this, and *effects* describe changes in the state of the world that result from the service execution. The functional service description of a Web Service applies the set-based modeling approach described above.

Similar to the Partner Discoverer, the architecture of the Service Discoverer relies on the relation of Goal Templates and Goal Instances as the following holds:

$$\text{If instanceOf}(GI_x, GT_x) \text{ then } \text{match}(GI_x, S_y) \Rightarrow \text{match}(GT_x, S_y) \quad (3)$$

This allows a layered architecture that consists of two subsequent building blocks as shown in Figure 6. The first one is the Pre-Selector that detects usable Services for Goal Templates. It is invoked at system setup time, i.e. whenever a new Goal Template or Service is deployed in the system; for each Goal Template GT, a set of matching Services $\{S_{GT}\}$ is determined as an intermediary discovery result. This is used as input for the GIS Matcher as the second building block. The GIS Matcher determines the Services that match the Goal Instance GI_i associated with the Discovery Request Q_{SD}; therefore, the intermediary discovery result $\{S_{GT\text{-}i}\}$ of the Goal Template GT_i is matched with GI_i.

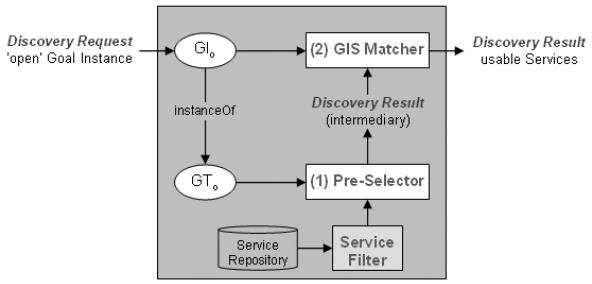

Figure 6: Service Discoverer Architecture

The Pre-Selector is comprised of two discovery functionalities. The first one is the Service Filter that selects services out of the Service Repository that have action equality with existing Goal Templates: $\{S_{filter}\}$ = *action-equality*(S, GT), based on the resources types defined in the action-resource ontology. Then, object matchmaking on postconditions and effects is performed on $\{S_{filter}\}$, deriving the intermediary discovery results $\{S_{GT}\}$ that contains all matching services for each existing Goal Template GT.

The GIS Matcher performs matchmaking of the invoking Goal Instance GI_i with $\{S_{GT\text{-}i}\}$ as those services of $\{S_{GT}\}$ that have been detected for GT_i corresponding GI_i. As action knowledge discovery is already completed within the Pre-Selector, the GIS Matcher only performs object matchmaking.

This is achieved in a two-step process. At first, the satisfiability of the precondition and assumption in the service description by the submission of the Goal Instance is investigated. These parts of the service description define conditions that have to hold before the service can be executed; they have to be satisfied by the concrete input provided to the Web Service, in our case the submission described in goal instances GI_i:

$$\forall x.\ GI_{submission}(x) \Rightarrow S_{precondition}(x) \wedge S_{assumption}(x) \quad (4)$$

Evaluating this satisfiability can result in three states: exact satisfaction, oversatisfaction, or undersatisfaction. While the former two do not affect service usability, for the latter the user is notified and can choose whether to submit additional information to complete the input needed for the Web Service or to cancel the service usage. If the satisfiability of precon-

ditions and assumptions is given, the GIS Matcher performs object matchmaking on postconditions and effects between GI_i and the respective services (for specific inputs that satisfy the precondition and assumptions), resulting in the final discovery result R_{SD} that contains those services that can be used by the goal owner for automated cooperation execution.

4.3. Testing

In order to test and evaluate the components for partner and service discovery, we defined an exhaustive use case in a virtual marketplace for purchasing furniture that contains several goals, services, and agents of marketplace participants. While the complete resource models and testing results are provided in [15], we summarize here the most relevant findings on usability and performance of the components.

The first important finding concerns the need for distinction of action and object knowledge as well as different discovery techniques for these. The use case contains several goals and services for buyers and sellers, which carry references to the respective actions in the action-resource ontology. When only performing object matchmaking, the discovery result contained resources that are not intended to be matching; for example, partner discovery for a ‘buyergoal’ detected other ‘buyergoals’ that satisfied object matchmaking; similar effects have been recognized within service discovery. Although the object definitions in the resources have been modeled correctly regarding the intended meaning, this emphasizes that action knowledge definitions along with compatibility notions is needed in order to ensure correct discovery results.

Regarding the performance of the discovery components, the proof obligations for object matchmaking with VAMPIRE have been found in 1-2 second maximum in average (testing environment: Linux Red Hat 9.0 on a Pentium 4 IBM machine); more complex proofs including convoluted axiomatic expressions can take significantly longer. We noticed that object structures without complex axiomatic definitions are most the common case in resource descriptions. Also, we regard correctness of matchmaking to be more important than celerity. In this respect, the performance can be considered as accurate with regard to the functional requirements. A prototype based on a deductive database system is an alternative with better performance characteristics. This is subject to future research.

As a final aspect we like to mention feedback from users that we gained throughout our project work. The users have mainly been system developers that usually work with conventional technologies and applied our technology for developing end-user applications. These users regarded exhaustive tool support as very important that is provided to applications developers as well as for end-users that 'hides' the logical aspects. For example, an end-user interface has been requested that allows creation and edition of goals without enforcing the user to formulate logical expressions; for developers, tool support has been requested that minimizes the effort for creation of resource description and deployment. Furthermore, abandonment of expressivity has been accepted in order to enhance usability. These requests show that 'conventional system developers' regard tool support and automation for system development and maintenance as very important; we mention this here explicitly in order to point attention and awareness to these issues that we believe are crucial for success and usage of semantic technologies.

5. Conclusions and Related Work

In this paper we have outlined an approach for automated collaboration support with Semantic Web Services and presented the theoretical basis as well as the realization of semantically driven partner and service discovery as core components for collaboration establishment. The following depicts the main outcomes of our work and discuss related work.

The first aspect to be addressed is the approach for collaboration support with Semantic Web Services, regarding expedient combination of the key technologies identified for the Semantic Web. To our understanding, our conceptual model attains an accurate assembly of technologies according for supporting automated collaboration on the Semantic Web. The main merit of the framework is that collaborations between entities is automatically executed by interaction of services; thereby, ontologies are used in order to ensure semantically correct information interchange, and other semantically described resources provide supplementary constructs that enable automated and high precision detection of potential collaborations. While several approaches for Web Service discovery rely on a requester-provider model, comparable conceptual models have been developed for collaborative multi-agent systems wherein autonomous agents interact via services, using ontologies as the semantic data model [9]. We believe that conceptual models like the one presented are appropriate for collaboration support on the Semantic Web as they support collaboration of symmetric partners and abstract from service usage models with restrictive roles.

The second aspect to be considered is the techniques for semantic discovery. Although the distinc-

tion of actions and objects as different types of knowledge that is used for describing Web Services and related resources has not been defined clearly yet, the idea of using resource classifications as filters for discovery has been supposed. For example, the METEOR-S project defines an OWL-S service profile hierarchy that classifies the types of services according to action knowledge and applies this as a pre-filter for discovery [11]. However, this approach does not explicitly model action knowledge and the relation to resources as within the action-resource ontology, and thus does not support inference-based filtering on more complex resource structures. Also, the functional need for applying both action and object knowledge discovery techniques has not been considered so far.

Concerning object matchmaking, several related approaches and techniques have been developed in recent research efforts. For example, [6] applies subsumption reasoning on basis of Description Logics for semantic service discovery, while [1] applies rewriting techniques in order to establish matching of a service usage request with OWL-S profiles. However, these techniques lack of generalized matchmaking notions as well as in precision of discovery results. The framework for Web Service Discovery in WSMO [8] that we have applied and extended relies on an exhaustive study of existing work and combines these into a concise theoretical basis for high precision object matching, wherefore VAMPIRE provides a suitable technical platform.

The final aspect to be mentioned is the appropriate combination of different discovery techniques as we have realized within the partner and service discoverer. In most of the above mentioned approaches for semantically enabled discovery, such combinations are favored; Also, layered architectures for Web Service discoverers are regard as the most appropriate solutions to ensure efficient handling of possibly numerous resources; in this respect, the architecture of the partner and service discovery components provide a prototypical solution for effective discoverer architectures. Furthermore, as it is expected that not all Semantic Web Services descriptions follow the same framework, collections of different techniques are considered for sophisticated support for Web Services.

Acknowledgements

The work presented here has been achieved within the Semantic Web Fred project (see project homepage: http://swf.deri.at/). We like to thank Peter Zugmann and Bernhard Keimel from Quarto Software, Vienna, for project collaboration, and the members of the WSMO working group for fruitful discussion and team work.

6. References

[1] B. Benatallah, M-S. Hacid, C. Rey, and F. Toumani: *Request rewriting-based web service discovery*. In The Semantic Web - ISWC 2003; pp. 242–257.

[2] Berners-Lee, T.; Hendler, J.; Lassila, O. (2001): *The Semantic Web. A new form of Web Content that is meaningful to computers will unleash a revolution of new possibilities.* In: Scientific American May 2001.

[3] R. Davis and R. G. Smith. *Negotiation as a metaphor for distributed problem solving*. Artificial Intelligence, 20(1):63ñ109, January 1983.

[4] de Bruijn, J, (Ed.): *The WSML Specification*. WSML Working Draft D16, 29 October 2004; www.wsmo.org/2004/d16/

[5] Fensel, D.: *Ontologies. A Silver Bullet for Knowledge Management and E-Commerce*. 2nd Edition. Berlin, Heidelberg: Springer, 2003.

[6] J. Gonzalez-Castillo, D. Trastour, C. Bartolini. *Description logics for matchmaking of services*. In Workshop on Applications of Description Logics, 2001.

[7] Li, L. and Horrocks, I.: *A software framework for matchmaking based on semantic web technology*. In Proceedings of the 12th International Conference on the World Wide Web, Budapest, Hungary, May 2003.

[8] Keller, U.; Lara, R.; Polleres, A. (eds.): *WSMO Web Service Discovery*. WSML Working draft D5.1, 12 November 2004; www.wsmo.org/2004/d5/d5.1/v0.1/

[9] Martin, D. L.; Cheyer, A. J.; Moran, D. B.: *The Open Agent Architecture: A Framework for Building Distributed Software Systems*. Applied Artificial Intelligence, vol. 13, no. 1-2, 1999; pp. 91-128.

[10] The OWL-S Coalition: *OWL-S 1.1 Release*, November 2004; available at: www.daml.org/services/owl-s/1.1/

[11] M. Paolucci, T. Kawamura, T. Payne, and K. Sycara: *Semantic matching of web services capabilities*. Proceedings of the First International Semantic Web Conference, Springer-Verlag, 2002; pp 333-347.

[12] Riazanov, A.; Voronkov, A.: *The design and implementation of VAMPIRE*. AI Communications 15(2), Special Issue on CASC, 2002; pp. 91 -110.

[13] Roman, D.; Lausen, H.; Keller, U. (eds.): *The Web Service Modeling Ontology WSMO.* WSMO Working Draft D2, final version 1.0, 20 September 2004; www.wsmo.org/2004/d2/v1.0/

[14] Stollberg, M.; Roman, D.; Toma, I.; Herzog, R.; Zugmann, P.; Fensel, D..: *Semantic Web Fred - Automated Goal Resolution on the Semantic Web.* In Proceedings of the 38th Hawaii International Conference on System Science (HICSS-38), January 2005.

[15] Stollberg, M. (ed.): *SWF Use Case*. WSMO Working Draft D3.5, 19 Oct 2004; www.wsmo.org/2004/d3/d3.5

[16] Tsarkov, D.; Riazanov, A.; Bechhofer, S.; Horrocks, I.: *Using Vampire to Reason with OWL*. In van Harmelen, F., McIlraith, S. and Plexousakis, D. (Eds.): Proceedings of the 3rd International Semantic Web Conference (ISWC 2004), 2004.