Trading Services in Ontology-driven Markets

Steffen Lamparter
Institute AIFB
University of Karlsruhe (TH)
Karlsruhe, Germany
lamparter@aifb.uni-karlsruhe.de

Björn Schnizler
Information Management and Systems
University of Karlsruhe (TH)
Karlsruhe, Germany
schnizler@iw.uni-karlsruhe.de

ABSTRACT

In order to realize the vision of a full-fletched service oriented architecture efficient service discovery and allocation is required to coordinate the interplay between service providers and requesters. This paper presents the architecture of an ontology-driven market for trading Semantic Web Services. An auction schema is enriched by a set of components enabling semantics based matching as well as price-based allocations. Moreover, an approach for reducing the complexity of the auction system by means of background knowledge is proposed.

Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Web-based Services*; K.4.4 [**Computers and Society**]: Electronic Commerce

Keywords

Auction, Web Services, Ontology

1. INTRODUCTION

Web Services have become the key technology for enabling distributed computing infrastructures for collaboration and electronic commerce. Although in theory, users could utilize Web Services from multiple locations, this potential is rarely exploited in practice. This seeming contradiction is caused by the significant barriers that arise when organizational boundaries are crossed. Overcoming these barriers requires that Web Services can be reliably discovered, acquired, and managed.

Most of the current service discovery mechanisms for Web Services use matchmaking algorithms which rely on attribute-based matching functions. However, these algorithms fall short of capturing the semantics required for automatic service discovery. Recently, the Semantic Web community suggested to enrich services by a formal and unambiguous description of their capabilities. Instead of syntactic matching algorithms, services can be semantically matched using concepts and relations formalized by means of ontologies.

SAC'06 April 23-27, 2006, Dijon, France

However, the direct application of semantic matchmaking mechanisms for allocating Web Services has several drawbacks: Firstly, these algorithms do not guarantee that those requesters will receive the supplied services who value them most. Secondly, they ignore the fact that users will only offer their services if they are adequately compensated. Compensation requires determining how the offered services are allocated among potential requesters and how the prices for the services are set. Thirdly, semantics based matchmaking algorithms are typically computational demanding and thus not adequate for large-scaled negotiations. However, these aspects are crucial for implementing economic efficient Web Service infrastructures.

Recently, researchers have suggested employing market mechanisms for the allocation of Web Services [8]. Markets can be an effective institution to allocate resources (Pareto-) optimally [10]. This is achieved by the interplay of demand and supply and due to the information feedback inherent to the price system. As such, the application of market mechanisms for the service discovery is deemed promising. However, applying classical market mechanisms for trading Web Services may lead to inefficient outcomes as these mechanisms allocate on base of syntactical descriptions. Thus, the goal of the paper is to enrich an auction mechanism with semantic matching capabilities.

Thus, the contribution of this paper is the design of a market platform for trading Web Services efficiently. This is realized by merging the advantages of classical auction algorithms and semantically based matchmakers into one framework.

The remainder of the paper is structured according to the main components that affect the design of a market platform [10]: (i) *A communication language* which defines how orders (i.e. offers and requests) and agreements can be formalized. (ii) *An outcome determination* by means of an allocation (i.e. who gets which service) and a pricing component. In section 2 general design requirements for a these components are elicited, before in section 3 an ontology based communication language and in section 4 the architecture of the outcome determination component is introduced. Section 5 compares the marketplace with related work. Finally, section 6 concludes with a short outlook.

2. DESIGN REQUIREMENTS

The platform should allow multiple buyers and sellers to *trade simultaneously* (R1) and ensure an *immediate execution* in case a suitable counterpart is found (R2). The mechanism should support the *trading of heterogenous services* like, for instance, a stock quote service and a billing service (R3). A *meaningful matchmaking of orders* should be realized by the market infrastructure to allow matching of services based on the semantics of an order instead of their syntactical representation (R4). For instance, a request for

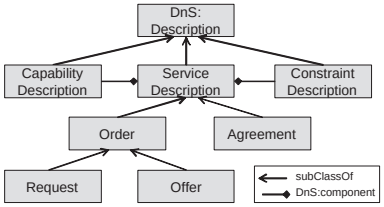

Figure 1: Market information as DnS:Description hierarchy.

a future stock quote service should be matched with an offer for a general stock quote service.

Additionally, the market should support the *trading of dependent services* (R5), as service may be complementarities. Participants have super-additive valuations for the resources, as the sum of the valuations for the single resources is less than the valuation for the whole bundle. Participants may also want to submit more than one bid on a bundle but many that are excluding each other. In this case, the resources of the bundles are substitutes due to sub-additive valuations for the resources. Furthermore, services can differ in their quality characteristics and their policies, e.g. a stock quote service by its quote time; a billing service by its age restriction. As such, the mechanism should support *multiattribute services* (R6).

A *computational and communication efficient determination of the outcome* (R7) is required by the mechanism in order to converge on a desirable global outcome by minimizing the computational effort.

3. ONTOLOGY-BASED COMMUNICATION LANGUAGE

In order to allow meaningful matchmaking (as postulated by R4) communication with the market has to take place on a semantic level. Ontologies provide the right means for specifying such semantics by featuring logic-based representation languages. In the following an ontology for the required communication primitives as well as service descriptions contained in these primitives is introduced. Thereby, we rely on the foundational ontology DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) [9]. Together with the extension Descriptions & Situations (DnS) [3], DOLCE can be regarded as a design pattern for ontologies. Moreover, we present an example for the approach.

Communication Primitives and Service Descriptions

The interaction between participants and the marketplace affects three communication primitives[1]: *Offers*, *Requests*, and *Agreements*. As depicted in figure 1, *Offers* and *Requests* are modeled as a special type of *Order*. In this context a *Request* is an *Order* that defines a maximal price of the service and an *Offer* is an *Order* containing a reservation price. *Orders* and *Agreements* can be regarded as a specialization of the concept *Service Description*, which is a subclass of the upper level concept *DnS:Description* derived from DnS.

A *Service Description* specifies concrete service capabilities and constraints, which are also modelled as *Descriptions* in terms of DnS. A *Service Description* must have at least one *Capability Description*, whereas constraints can be defined optionally. These two components are described in the next section.

In literature several explicit as well as implicit formal modeling approaches for service capabilities exist [12]. For our work we reuse a revised version of the Core Ontology of Services [4] which is basically suited to model both approaches mentioned above. For simplification, we use a pure implicit modelling approach, where

[1]Concepts and associations of the ontology are written in italics. Those derived from DOLCE and DnS are labelled with the according namespace.

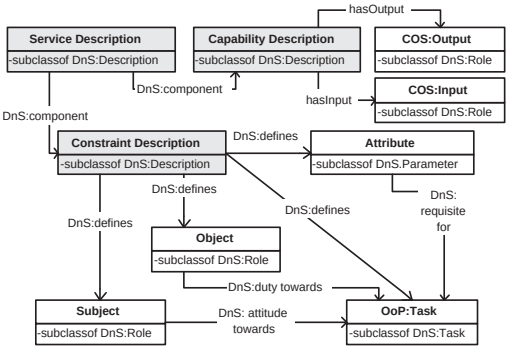

Figure 2: Sketch of the service description ontology.

only inputs and outputs are considered for capability matching in the marketplace. As shown in figure 2, a *Capability Description* consists of *Inputs* as well as *Outputs*. These descriptive entities are *Roles* that are played by *Information Objects*, e.g. a concrete message in the system. In order to allow bidding on service bundles, service descriptions may include more than one capability description.

Besides service capabilities service providers and requesters may want to express additional requirements that have to be fulfilled. Such requirements could be constraints regarding the desired/offered service or constraints regarding properties of providers and customers themselves, e.g. the age of a customer must be at least 18 years. In the context of Web Services these constraints are often referred to as policies. For modeling of such constraints the framework presented in [5] is used. As sketched in figure 2, a *Constraint-Description* consists of the concepts *Subject*, *Computational Task*, *Object*, and *Attribute*. Different modalities can be expressed by specializing the *DnS:attitudeTowards* relation between *Subject* and *Task*, e.g. *DnS:rightTowards*, *DnS:obligedTo*. By means of the the concept *Attribute* non-functional properties of a service can be constrained. *Attribute* is *valued-by* a DOLCE *Region* that defines the valid range of the attribute values.

Example

As an example consider financial Web Services returning stock quotes. Capabilities defining the input and output of such services are shown in figure 3. Typically such services are not for free and the prices charged by the different providers vary over time. In their offer providers may specify a minimal price for which they are willing to offer the service. Customers state the maximal price they are willing to pay in their request. The final price of the transaction is determined by supply and demand in the market.

For instance, a provider A is supplying a service capable of returning quotes in Dollar for any given security (i.e. options, shares, and futures). The provider has a reservation price of 5 Dollars for this service. In the following the central parts of the offer are sketched by means of description logic axioms [1]:

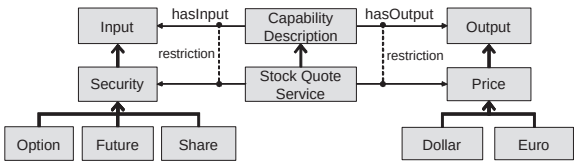

Figure 3: Example Stock Quote Services

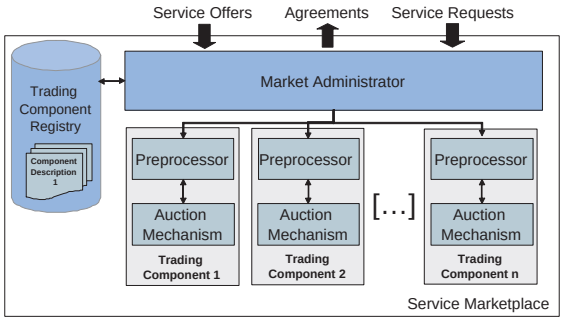

Figure 4: Overall architecture of the service marketplace

ServiceA $\sqsubseteq$ Offer $\sqcap \forall$DnS:component.CapA $\sqcap \exists$DnS:component.ConA
CapA $\sqsubseteq$ CapabilityDescription $\sqcap \forall$hasOutput.Dollar $\sqcap \forall$hasInput.Security
ConA $\sqsubseteq$ ConstraintDescription $\sqcap \exists$defines.ReservationPriceA
ReservationPriceA $\sqsubseteq$ Attribute $\sqcap \forall$DnS:valuedBy.$\{\geq 5\}$

A customer B is demanding a stock quote service capable of providing quotes in dollar for futures. B valuates this service with 6 Dollars. This results in the following (simplified) request:

ServiceB $\sqsubseteq$ Request $\sqcap \forall$DnS:component.CapB $\sqcap \exists$DnS:component.ConB
CapB $\sqsubseteq$ CapabilityDescription $\sqcap \forall$hasOutput.Dollar $\sqcap \forall$hasInput.Future
ConB $\sqsubseteq$ ConstraintDescription $\sqcap \exists$defines.ValuationB
ValuationB $\sqsubseteq$ Attribute $\sqcap \forall$DnS:valuedBy.$\{\leq 6\}$

4. MARKETPLACE ARCHITECTURE

Having defined the communication language, this section tailors the overall architecture of an ontology-driven marketplace. The architecture, as outlined in figure 4, consists of the following components: a Marketplace Administrator, a Trading Component Registry, and several instances of Trading Components which encapsulate Preprocessors and Auction instances.

Although the market is seen as a monolithic unit, internally it may consist of several independent trading components each with its own auction mechanism and preprocessors. In each of these individual trading components, only a subset of all available services is traded. Although a central auction mechanism comprising all available services would fulfill the economic properties, a distinction between several independent trading components reduces the complexity of the auction (R7) since the allocation time increases exponentially with respect to the number of bids in the order book. In the following the components of the marketplace architecture are described in detail.

4.1 Trading Component Registry

The Component Registry is a repository that contains an ontology-based description of each trading component. These *Component Descriptions* basically specify the capabilities of the services that are traded in a specific trading component. From a modeling perspective they can also be regarded as a specialization of the concept *Service Description* introduced in section 3. In order to allow forwarding of incoming orders to the right component, it has to be described by the most general inputs and outputs. For instance, a component trading the orders mentioned in example 3 is described by the input *Security Paper* and the output *Dollar*. Note that a component description may comprise several capability descriptions.

4.2 Marketplace Administrator

The Marketplace Administrator manages the internal mechanisms, i.e. the administrator creates, removes, splits, or merges trading components based on the ontological description of the orders they contain. Moreover, the administrator receives orders from market participants and forwards these orders to the appropriate trading component. This is done by comparing the semantical description of the orders with the description of the trading components contained in the registry (R4).

Algorithm 1 ProcessIncomingOrder(O)

Set of related descriptions $\mathcal{M} = \emptyset$
for all capabilities C_k^O in the incoming order O **do**
  for all trading component descriptions tc_i **do**
    for all capabilities C_{ij}^{Desc} specified in the description of tc_i **do**
      if RELATED(C_k^O, C_{ij}^{Desc}) **then**
        $\mathcal{M} = \mathcal{M} \cup \{tc_i\}$ to $\mathcal{M}$
if $\mathcal{M} = \emptyset$ **then**
  CREATE new trading component for O
else $\{|\mathcal{M}| = 1\}$
  FORWARD(tc) with $tc \in \mathcal{M}$
else
  MERGE those components where the description is contained in $\mathcal{M}$

Algorithm 2 RELATED(Capability C^1, Capability C^2)

$O_{C^1} = hasOutput^{-1}.C^1$, $I_{C^1} = hasInput^{-1}.C^1$
$O_{C^2} = hasOutput^{-1}.C^2$, $I_{C^2} = hasInput^{-1}.C^2$
return $(O_{C^1} \sqsubseteq O_{C^2} \vee O_{C^2} \sqsubseteq O_{C^1}) \wedge (I_{C^1} \sqsubseteq I_{C^2} \vee I_{C^2} \sqsubseteq I_{C^1})$

In order to manage the trading components we have to distinguish between two events: (i) A new order may come into the market or (ii) an agreement is closed and thus orders are removed from the market. In each case different operations have to be carried out by the Market Administrator:

Incoming Order: Once an order is received, the administrator checks whether there exists already a suitable trading component (TC). This is done by comparing the capabilities of the component descriptions derived from the registry and the capabilities contained in the incoming order. Algorithm 1 illustrates this approach[2]. Capabilities are *related* if there is a subsumption relation between the inputs and the outputs (refer to algorithm 2). If there is one pair of capabilities with such a relation an order matches a trading component description. As a result one of the following operations are applied:

FORWARD to TC: If exactly one trading component is related to the order, the order will be forwarded to this component. Besides an update of the component description no additional operations are required.

CREATE TC: If no existing trading component is related to the incoming order a new component has to be instantiated. The incoming order has to be forwarded to that component and a new component description has to be added to the registry.

MERGE $TC_1, \ldots, TC_n$: Trading components are merged by integrating the corresponding order books and updating the component description. Merging becomes necessary when a new order arrives that is related to capabilities traded in different components. Especially since the market allows dependencies between orders (R5) which can be used to relate entirely different services. Suppose there are two trading components, one dealing with stock quote and one with currency translation services. If a request for a financial service in general arrives, these two markets have to be merged. Furthermore, all services requested or offered in a bundle order have to be traded in one single component.

Removing Order: Each time an order is removed from the market several inconsistencies may arise: Firstly, there might be trading

[2]In the following algorithms $X \sqsubseteq Y$ means that Y subsumes X. Moreover, rel^{-1} refers to the inverse relation of rel.

components which are not required any more. Secondly, splitting of components might be possible since orders requiring a merged trading component might have been removed from the market. Therefore, the following two operations are required:

DELETE TC: A trading component has to be removed together with the according component description in the registry if the order book is empty, i.e. no orders with the according capabilities remain in the market.

SPLIT TC: Trading components have to be split in case they contain two fully distinct sets of service types. Therefore, the split operation does not lead to any market defects (e.g. missed matches, decreasing liquidity). Imagine a trading component contains stock quote as well as billing services. Obviously, the order book can be split into one book containing only stock quote and one containing only translation services without loosing potential matches. Trading components should be divided as soon as possible to reduce the complexity of the allocation mechanism.

4.3 Trading Component

Each instance of a Trading Component consists of two building blocks: An Auction and a Preprocessor component. In order to compute the optimal outcome, a multiattribute combinatorial double auction is applied [11]. This mechanism meets the requirements R1, R2, R3, R5, and R6 mentioned in the last section. However, existing implementations of this mechanism rely only on pure syntactic matching of orders and thus requirement R4 cannot be met. Therefore, an additional preprocessing of the order book is necessary.

Preprocessor

The Preprocessor administrates an order book containing ontology based orders and is responsible for preparing this order book in a way that it can be handled by a traditional auction system where bids are represented by strings without formal semantics. Thereby, two aspects can be distinguished: ***(i)*** *Semantic preprocessing* where ontological information is used to introduce new XOR-orders and ***(ii)*** *syntactic preprocessing* where the ontology based communication primitives are translated into the syntactic bidding language that is understood by the auction mechanism. In the following these two steps are described in more detail.

Semantic Preprocessing: An offer *semantically matches* a request (R4) if all inputs and outputs match. Inputs match in case all inputs of the offer subsume at least one input of the request. Outputs match in case all outputs of the request subsume at least one output of the offer. The rational behind this approach is to make sure that the service understands all possible inputs sent by the requester and that the requester is able to handle all possible answers received from the service. The approach also reflects the fact that services which need less inputs and provide more outputs than required by the requester are also suitable matches.[3] However, the objective of the preprocessing component is not to match orders directly, but to transform the orders in a way that the subsequent syntactic auction mechanism results in the same set of matches as a semantic approach, while inappropriate allocations are avoided. *Syntactic matches* are realized in case the inputs as well as the outputs contained in the corresponding request and offer are identical, i.e. they are described by the same concept.

Based on these definitions, the orders mentioned in example 3 match only on a semantic level due to the different degree of abstraction of the input concepts. In order to get the same matches in a syntactic algorithm as in a semantic approach, the ontology is used to generate alternative bids. These bids can then be concatenated by a XOR operator, ensuring that at most one order will be executed. Therefore, the following rules can be used (algorithm 3):

[3]Free-disposal is assumed, i.e. the bidder always accepts better services for free.

Algorithm 3 Semantic preprocessing of orders

for all orders in the trading component **do**
  for all concepts I_i^R representing the inputs of a request R **do**
    for all input concepts $I_k^{Onto} \neq I_i^R$ in the ontology **do**
      if $I_i^R \sqsubseteq I_k^{Onto}$ and $I_k^{Onto} \sqsubseteq I^{Desc}$, where I^{Desc} is an input in the component description **then**
        add an XOR request with I_k^{Onto} and the outputs as in R
  for all concepts O_i^O representing the outputs of an offer O **do**
    for all all output concepts $O_k^{Onto} \neq O_i^O$ in the ontology **do**
      if $O_i^O \sqsubseteq O_k^{Onto}$ and $O_k^{Onto} \sqsubseteq O^{Desc}$, where O^{Desc} is an output in the component description **then**
        add an XOR offer with O_k^{Onto} and the same inputs as in O

If a concept X that represents the input of a request (e.g. Future) is subsumed by another concept Y in the ontology (e.g. Security), a new XOR-associated request is introduced with input Y. This has to be done for each concept that subsumes X and is subsumed by the input of the component description. This means for request B of the example 3 that the alternative request (Security, Dollar) is introduced. Analogously, on the offer side orders with more general outputs are introduced. Thus, for offer A of the example, a new offer (Security, Price) is added. Recapitulating, we get two alternative requests (Security, Dollar) and (Future, Dollar) as well as two offers (Security, Price) and (Security, Dollar). Now, it is easy to see that there will also be a match in the syntactic approach. The auction mechanism has to be able to guarantee that only one alternative request and offer is accepted (XOR-bids).

In case of semantic matching, it can be shown that after applying these transformation rules the possible matches are identical compared to the situation without applying the rules. In case of syntactic matching, applying the rules could significantly improve the matchmaking (as illustrated in the example), while it is guaranteed that at least the same matches are found compared to situation without applying the rules.

Syntactic Preprocessing: After the semantic preprocessing, the ontological orders are translated into the syntactic bidding language of the auction mechanism. For the auction mechanism, a buyer order has to be transformed into a XOR concatenated set of bundle bids $B_{n,1}(S_1) \oplus \ldots \oplus B_{n,u}(S_j)$, where $n \in \mathcal{N}$ is an arbitrary buyer, S_i is a bundle (e.g. a stock quote service and a weather service) and u is the number of bundle bids in the order. A single bundle bid $B_{n,f}(S_i)$ is defined as the tuple

$$B_{n,f}(S_i) = (v_n(S_i), (q_n(S_i, g_1, a_{g_1,1}), \ldots, q_n(S_i, g_j, a_{g_j,A_j}))).$$

In the bundle bid, $v_n(S_i)$ defines the valuation for this service, i.e. the highest price, a buyer is willing to pay. Furthermore, the attribute characteristics can be expressed by $q_n(S_i, g_i, a_{g_j,k})$ where $g_i \in S_i$ is a specific service and $a_{g_j,k} \in g_i$ is an attribute of it. The orders of the sellers $m \in \mathcal{M}$ are formalized in a similar way as the buyers' orders are; the valuation $v_n(S_i)$ is replaced by a reservation price $r_m(S_i)$, i.e. the minimum price, a seller wants to get for the service.

To illustrate the preprocessor, suppose the following transformation: The requests (Security, Dollar) and (Future, Dollar) are represented as two XOR concatenated bids $B_{m,1}(SecPaper) \oplus B_{m,2}(Future)$ with $B_{m,1}(SecPaper) = (6, Dollar)$ and $B_{m,2}(Future) = (6, Dollar)$. The XOR operator ensures, that at most one bid will be executed. The offers will be transformed

analogously to $B_{n,1}(SecPaper) \oplus B_{n,2}(SecPaper)$ with $B_{n,1}$ $(SecPaper)= (5, Dollar)$ and $B_{n,2}(SecPaper) = (5, Dollar)$.

Combinatorial Double Auction

The auction mechanism used in the architecture is a multiattribute combinatorial exchange (MACE) as proposed in [11]. MACE allows multiple buyers and sellers simultaneously (cf. R2 in section 2) the submission of bids on heterogonous services expressing (R3,R5) substitutabilities (realized by XOR bids) and complementarities (realized by bundle bids).[4] Furthermore, the mechanism is capable of handling cardinal attributes (R6) as well as an immediate execution of given orders as the clearing can be done continuously (R2). Besides a syntactically based bidding language, the auction contains a winner determination component (i.e. allocation of services from suppliers to requesters) and a pricing schema (i.e. which price have to be paid on base of the allocation). In [6] it is shown that the presented architecture reduces the overall complexity of the used auction schema.

5. RELATED WORK

For maintaining a loose coupling between service requester and provider, dynamic service discovery plays a crucial role. Thus, several algorithms and frameworks have been proposed to tackle this problem. Some of them are based on syntactic service descriptions, like description repository UDDI or the discovery protocol WS-Discovery and feature symmetric and attribute-based matching of service descriptions. This is inflexible and difficult to extend to new characteristics or concepts. Others, like [12, 2, 7], suggest to adopt semantic service descriptions for matchmaking. However, while providing semantic matching capabilities, these algorithms use centralized matching components without the employment of prices. Thus, these algorithms require full information about the demand and supply situation in order to be effective. However, this information requirement is not even closely met [11].

Market-based approaches incorporate incentives for truthful information revelation by implementing prices. [8] motivate exemplarily the use of price systems for allocating Web Services. However, complementarities, attribute characteristics, and semantics based order specifications are not considered and, thus, the mechanism does not fulfill the requirements defined in section 2.

Furthermore, [13] introduces a ontology-based framework for matchmaking and negotiation of e-services. However, they do not present a concrete negotiation mechanism.

6. CONCLUSION

This paper outlined the design of an ontology-driven market for trading Web Services. Based upon a requirement analysis for a Web Service market, a marketplace was designed which is up to these marks. The marketplace uses an ontology based communication language that is capable of representing semantically described request, offers, and agreements. These ontology-driven messages were transformed into syntactically represented orders so that an existing auction mechanism could be used while still allocating on a semantic level. Furthermore, semantic information was used to split the whole market into several independent sub-markets. The concept was shown to be more efficient than an existing mechanism, i.e. the use of background knowledge has reduced the overall complexity.

For the future, more sophisticated service descriptions as well as additional auction mechanisms (e.g. English auctions) will be integrated for making the overall platform generally applicable. Furthermore, the existing platform will be compared and benchmarked with semantics-based matchmakers as well as classical auction marketplaces.

[4]It is to note, that bidders do not have to submit bids on all possible bundle combinations.

Acknowledgments. This work was funded by the European Union under the IST projects SEKT and CATNETS and the German Research Foundation (DFG) in scope of Graduate School Information Management and Market Engineering.

7. REFERENCES

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory Implemenation and Applications*. Cambridge University Press, 2003.

[2] S. Colucci, T. D. Noia, E. D. Sciascio, F. Donini, and M. Mongiello. Concept abduction and contraction for semantic-based discovery of matches and negotiation spaces in an e-marketplace. *Electronic Commerce Research and Applications*, 4(3), 2005.

[3] A. Gangemi and P. Mika. Understanding the semantic web through descriptions and situations. In *Confederated Int. Conf. DOA, CoopIS and ODBASE*, LNCS 2888. Springer Verlag, 2003.

[4] A. Gangemi, P. Mika, M. Sabou, and D. Oberle. An ontology of services and serivce descriptions. Technical report, Laboratory for Apllied Ontology, Rome, Italy, 2003.

[5] S. Lamparter, A. Eberhart, and D. Oberle. Approximating service utility from policies and value function patterns. In *6th IEEE Int. Workshop on Policies for Distributed Systems and Networks*. IEEE Computer Society, 2005.

[6] S. Lamparter and B. Schnizler. Trading Services in Ontology-driven Markets. Technical report, University of Karlsruhe (TH), 2005. `http://www.aifb.uni-karlsruhe.de/WBS/sla/paper/TRTradingServices.pdf`.

[7] L. Li and I. Horrocks. A software framework for matchmaking based on semantic web technology. In *WWW '03: Proceedings of the twelfth international conference on World Wide Web*, pages 331–339. ACM Press, 2003.

[8] Z. Li, H. Zhao, and S. Ramanathan. Pricing web services for optimizing resource allocation an implementation scheme. In *Proc. of the Web2003, Seattle, WA*, 2003.

[9] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, A. Oltramari, and L. Schneider. The WonderWeb library of foundational ontologies. WonderWeb Deliverable D17, Aug 2002.

[10] D. Neumann. *Market Engineering - A Structured Design Process for Electronic Markets*. PhD thesis, Economics and Business Engineering, University of Karlsruhe (TH), 2004.

[11] B. Schnizler, D. Neumann, D. Veit, and C. Weinhardt. A Multiattribute Combinatorial Exchange for Trading Grid Resources. In *Proceedings of the Research Symposium on Emerging Electronic Markets (RSEEM), Amsterdam, Netherlands*, 2005.

[12] K. Sycara, M. Paolucci, A. Ankolekar, and N. Srinivasan. Automated Discovery, Interaction and Composition of Semantic Web Services. *Journal of Web Semantics*, 1(1), 2003.

[13] D. Trastour, C. Bartolini, and C. Preist. Semantic web support for the business-to-business e-commerce pre-contractual lifecycle. *Computer Networks*, 42(5):661–673, 2003.