Specification of Policies for Automatic Negotiations of Web Services

Steffen Lamparter and Sudhir Agarwal

Institute of Applied Informatics and Formal Description Methods (AIFB),
University of Karlsruhe (TH), Germany
{lamparter,agarwal}@aifb.uni-karlsruhe.de

Abstract. Market mechanisms provide an efficient institution for allocating service offers and requests. In doing so, negotiations between the market participants play a crucial role. However, current policy languages are ill-suited to realize beneficial trade-offs within a negotiation, since they support only boolean decisions. Therefore, we suggest an approach where preferences are modeled as utility functions. We show, how such preferences can be specified with description logics to enable the use of existing inference engines for calculating the degree of policy satisfaction by offers/requests which can be considered in the negotiation process.

1 Introduction

Web services are self-contained, modular business applications that have open, Internet-oriented, standards-based interfaces, e.g. WSDL. They allow flexible and dynamic software integration that is often referred to as the ”Find-Bind-Execute”-paradigm. Moreover, by using standard Internet technology, Web services facilitate cross-organizational transactions and thus outsourcing of software functionality to external service providers. When moving from distributed systems operating within one company to systems that involve different, independent companies, the ”Find-Bind-Execute”-schema describes nothing else than a B2B procurement process, where digital services such as information delivery or execution of calculations are purchased. Thus, service-oriented computing requires an infrastructure that provides a mechanism for coordinating between service requesters and providers. This coordination mechanism has to provide a platform where potential business partners can be discovered, prices can be ascertained, and contracts can be closed. A marketplace, where prices are determined by the interplay between supply and demand, can be regarded as a coordination mechanism that efficiently provides these functionalities [1].

1.1 Web Service Markets

Figure 1 brings together the phases that can be identified in an electronic market [2, 3] and the typical Web Service usage process which comprises the steps discovery, composition, negotiation, and finally contracting. In the *Matchmaking Phase* suitable services are discovered. Since a certain goal can not be accomplished only by a single service but also by a combination of services this phase also includes composition. After having determined those services that are able to achieve a certain goal an optimal assignment of service requests and offers with respect to the individual utility of the participants or

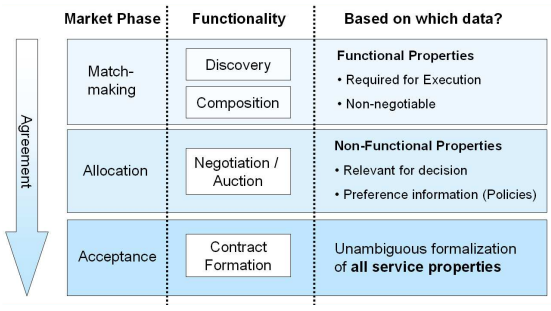

Fig. 1. Market phases and the Web Service usage process

to the overall welfare has to be found in the *Allocation Phase*. To achieve this, negotiations between the participants have to be carried out. For determining the allocation and price many different mechanisms are available ranging from simple selection approaches to complex negotiation or auction schemes. After this allocation, legally binding contracts are closed between the corresponding business partners in the *Contract Formation Phase*. These contracts have to be formalized in a machine-understandable way in order to allow automated execution and monitoring.

In each market phase different kind of information is required. *Functional properties*, required in the *Matchmaking Phase*, are those attributes that are mandatory to be able to invoke a service and to integrate the results, e.g. the input and output of a service. That means, that for functional properties no alternatives can be specified and thus negotiations about such properties are impossible. All discovered services fulfill the desired goal but may differ in their *non-functional properties* which are attributes that are not required to invoke the service nor to integrate the results, but they are the decisive factors for service selection and price determination. For example, price, payment method, security as well as trust attributes, and most notably quality of service attributes. Typically, for each non-functional attribute there are several alternatives that can be adopted depending on the preferences of the trading partners. Thus, a negotiation has to be carried out to agree on one of the alternatives. In order to be able to negotiate, preference information about the different alternatives is required. In case of an automatic negotiation it is not enough that preferences are in the user's mind, but they have to be formalized explicitly.

Here, the policies come into play. Policies allow to declaratively express preferences, i.e. which of the different alternatives a non-functional property may adopt. Thus, policies can be regarded as constraints or rules that restrict the decision space within the negotiation of agreements.

1.2 Some Motivating Scenarios

In this section some motivating examples are presented in order to illustrate why negotiations about non-functional properties are required. We come up with examples from the domains privacy and quality of service. However, the problems are the same for other non-functional properties.

Privacy. A Web service might support different privacy levels. For example, a provider either gives a guarantee to delete customer data straight after the business interaction was carried out or the provider stores customer data for further usage. In the latter case a discount on the service price is given to the customer, i.e. the customer could sell private data in exchange for a discount. Which of the alternatives is more preferable to the customer depends on how important data privacy as well as a cheap price is judged by the customer.

Quality of Service. A Web Service interaction involves several different quality of service criteria like response time, availability, etc. Typically, not all criteria are perfectly met by the service providers, rather each provider has his strengths and weaknesses. In order to decide which service suits best exact information about the requesters preferences are required, e.g. is a service with fast response time and bad availability better than a service with the converse properties. Moreover, one has to know if a $10 discount in price justifies a slower response time of 10s.

In each of this examples there is a trade-off between different service properties (e.g. quality vs. price, privacy vs. price, privacy vs. quality) which can only be resolved by making the different attributes comparable. This can be realized by assigning utility values to the different decision alternatives. Such cardinal preferences allow to decide whether a certain discount is high enough to compensate the loss in utility that results from a disadvantageous property value. Hence, negotiations between the parties may lead to service configurations that yield higher welfare for providers as well as requesters.

The paper is organized as follows. In section 2 a general policy framework is introduced and extended to enable the representation of fine-grained preferences. We show how preferences can be evaluated in a DL reasoner by mapping utility functions to an appropriate description logic. We conclude in section 4 after discussing related work in section 3.

2 Specifying Policies for Negotiation

In this section, a formalism is presented that allows formal specification of preferences and thus facilitates automatic decision making and negotiations. In section 2.1, a general framework for expressing policies is introduced. This framework is based on the foundational ontology DOLCE [4]. Foundational ontologies capture typical *ontology design patterns* (e.g. location in space and time). By providing a sound conceptual model with precise concept definitions they facilitate integration of different policy efforts (cf. [5]). In section 2.2 the description framework is extended to enable the representation of fine-grained preferences by means of description logics.

2.1 Policy Description Framework

In this section, the generic policy description framework introduced in [5] is refined. The framework provides a generic ontology for expressing policies. Ontologies formalize concepts and concept relationships very similar to conceptual database schemata or UML class diagrams [6]. However, ontologies typically feature logic-based representation languages. Those languages come with executable calculi that allow querying and

reasoning during run-time. Moreover, ontologies facilitate the conceptual integration of heterogeneous policy efforts by providing well-defined and machine understandable semantics.

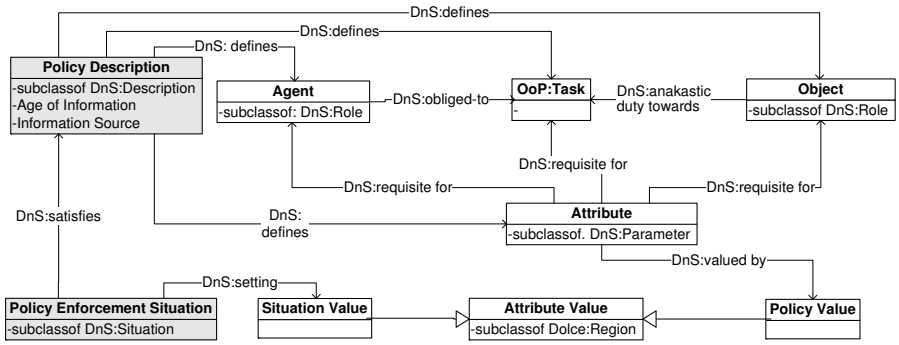

Fig. 2. Sketch of the Policy Description Framework.

In order to express policies we add a Core Policy Ontology to the DOLCE ontology stack. In the remaining part of this section we introduce the basic principles of DOLCE, present the design of the Core Policy Ontology, and show how the framework can be used to express concrete policies by means of an example.

DOLCE. The foundational ontology DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) provides the basis for the policy description framework used in this paper. Foundational ontologies are high-quality formalizations of domain independent concepts and associations that contain a rich axiomatization of their vocabulary. D&S (DnS) is an ontology module that extends DOLCE and introduces the basic distinction between descriptive (DnS : Description)[1] and ground entities (DnS : Situation). A Situation defines a state of affair (e.g. real settings in the world such as facts or cases), while a Description is a conceptualization which encompasses objects such as laws, plans, policies, etc. A detailed description of DOLCE and D&S can be found in [4] and [7], respectively. Moreover, in order to model workflow information as well as data the modules Ontology of Plans (OoP) and Ontology of Information Objects are introduced [7]. Descriptions contain Concepts such as FunctionalRoles, CourseofEvents, and Parameters. Ground entities in D&S are derived from DOLCE. FunctionalRoles are played − by Endurants, CoursesofEvents sequences Perdurants, Parameters are valued − by Regions.

Core Policy Ontology. In order to express policies we have to extend the basic vocabulary with policy specific concepts and relations, while reusing the foundational ontologies as far as possible. This core ontology contains the basic building blocks needed for modeling policies.

[1] Concepts of the ontology are written in sansserif. For concepts and relations that are directly contained in the corresponding ontology name spaces are omitted, for those derived from other modules the corresponding name space is mentioned explicitly.

Figure 2 sketches the Core Policy Ontology (CPO) in a simplified way. All concepts of the CPO are subclasses of DOLCE top-level concepts. A policy description consists of the concepts Agent, OoP : Task, Object, and Attribute. The entities Agent, OoP : Task, and Object allow to define the application area of the policy, while Attribute defines the property that is constrained by the policy. This could be, for instance, a constraint regarding the service, the agent that invokes the service, etc. The Attribute is DnS : valued − by an AttributeValue which is a Dolce : Region and specifies which attribute values are allowed according to the policy.

During run-time the policy has to be enforced by the system. This is done in a concrete PolicyEnforcementSituation which represents the current state of the system. In doing so, it has to be checked if the DnS : Concepts in the DnS : Description are DnS : classified by an entity in the DnS : Situation. If this is the case a DnS : satisfies relation is introduced between PolicyEnforcementSituation and PolicyDescription as specified in [8]. The actual attribute value in the situation (denoted by SituationValue) classifies the PolicyValue only in case the Dolce : Region defined in the SituationValue is contained in the Dolce : Region of the PolicyValue. In this case the policy is met.

Example. In the following we show how the framework introduced above is applied to specify concrete policies. Again we fall back on the privacy and quality of service domains. Consider a requester policy which says that a provider may store private data of the customer only for up to 14 days. This can be formalized by means of the Core Policy Ontology as show in figure 3. All concepts of the description are instantiated by domain specific entities. The policy will be applied if Web Service Providers (WSProvider) store PrivateData and permits this only for 14 days. Additionally, a quality of service policy is added, which specifies that the response time of the other party should be less than 5 seconds. Therefore, a additional Attribute ResponseTime is added. Both attributes are valuedby an IntegerRegion.

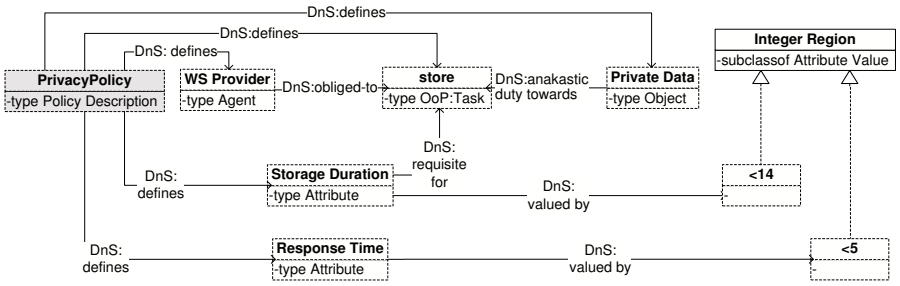

Fig. 3. Specification of a Privacy Policy.

2.2 Utility-based preference specification

In section 2 a generic framework for specifying policies is introduced. However, this framework is not expressive enough to capture fine-grained preferences as required for

supporting automatic negotiations. In the field of economics multi-attributive utility theory [9] is typically used to address these problems. It allows to handle trade-offs between alternatives and provides the right means for finding optimal service configurations. After introducing the basic idea behind utility theory, we show how such an utility approach can be integrated into our policy description model. To achieve this, utility functions are mapped to description logics. Further, we come up with an example to illustrate how fine-grained policies can be specified and rankings can be derived.

Utility Theory. In the context of utility theory a preference structure is defined by the complete, transitive, and reflexive relation $\succeq$. This means the property value $p_1 \in \mathcal{P}$ is preferred to $p_2 \in \mathcal{P}$ if $p_1 \succeq p_2$. The preference structure can be derived from the value function $v^i(p)$ of a user i.

$$\forall a, b \in \mathcal{P} : p_a \succeq p_b \Leftrightarrow v^i(a) \geq v^i(b)$$

The function $v^i(x)$ represents the utility defined by the relation $\succeq$ in a sense that the attribute values can be ranked by comparing the numeric values of the value function. Utility theory allows to decompose complex outcome spaces into utility functions composed of several lower-dimension functions. Thus, we can describe the preference structure for the attributes relevant to a specific service separately and then combine them to get the overall valuation. According to those definitions a user i specifies the utility function of the individual service properties X. Then, the overall valuation can be approximated by using the following additive value function.

$$V^i(x) = \sum_{j=1}^{n} \lambda_j^i v_j^i(x_j) \tag{1}$$

For the additive value function above we assume mutual preferential independence between the attributes [9]. Under this assumption we can easily aggregate the utility functions $v_j^i(x_j)$ of the individual attributes j to obtain the overall valuation of a service. Additive value functions are valid in many real world scenarios and might still provide a good approximation, even when mutual preferential independence does not hold exactly [10]. The weighting factor λ_j^i is normalized in the range $[0, 1]$ and allows to model the relative importance of an attribute j for a specific agent i.

Formal representation of preferences. In order to allow standard DL reasoners to make decisions based on the introduced utility approach, utility information has to be specified in a formal way. This can be done by modifying the concept PolicyValue of the Core Policy Ontology in a way that each property value in the set refers to a specific utility value. To allow for handling of discrete as well as continuous properties complex functions are required that map properties to utility values. In doing so, the satisfies-relation does no longer lead only to a pure boolean statement about the conformity of a Situation with respect to the PolicyDescription, but it leads to a statement about the degree of conformity. This is exactly the information that is required in order to automate negotiations. To facilitate the representation using description logics we restrict ourself to piecewise linear utility functions since such functions can be defined just by sets of points in $\mathbb{R}^2$. We use the description logic $\mathcal{ALC}(\mathcal{D} + \Sigma)$ with concrete domains and aggregates as proposed in [11]. The set of two points with adjacent x-coordinates

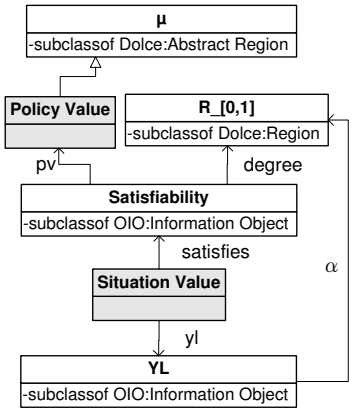

Fig. 4. Extended policy description framework.

can be interpreted as a straight line. For every line between (x_1, y_1) and (x_2, y_2) and a given x, we calculate an α as follows.

$$\alpha = \begin{cases} \frac{y_1 - y_2}{x_1 - x_2}(x - x_1) + y_1, \text{ if } x_1 \leq x \leq x_2 \\ 0, \text{ otherwise,} \end{cases} \tag{2}$$

We model this by defining a concept YL, to capture the αs as described above.

$$\mathsf{YL} \sqsubseteq \mathsf{OIO} : \mathsf{InformationObject} \sqcap \exists \alpha.\mathbb{R}_{[0,1]}, \tag{3}$$

where α is a functional role. This means for defining piecewise linear functions the semantics of PolicyValue has to be modified to a subclass of the Dolce : AbstractRegion μ that contains the set of points (x, y) which constitutes the utility function. Moreover, we define the relation $\exists$yl.YL from SituationValue to the Dolce : AbstractRegion YL. A SituationValue will be in as many relation instances of yl with instances of YL as there are lines in the utility function μ. The utility value of a SituationValue a according to μ is then just the sum of all such α over all the lines of μ. Further, we define a relation satisfies (specialization of OIO : realizes) from SituationValue to the OIO : InformationObject Satisfiability, which is defined as

$$\mathsf{Satisfiability} \sqsubseteq \mathsf{OIO} : \mathsf{InformationObject} \sqcap \exists \mathsf{pv}.\mathsf{PolicyValue} \sqcap \exists \mathsf{degree}.\mathbb{R}_{[0,1]}. \tag{4}$$

Now, the axiom

$$P_{=}(\mathsf{satisfies} \circ \mathsf{degree}, \sum(\mathsf{yl} \circ \alpha)), \tag{5}$$

where the predicate $P_{=}(x, y)$ is true iff $x = y$, ensures that the utility value of an individual a according to the function μ is equal to the sum of all α over all lines of μ. Based on this result we can calculated the weighted degree of satisfaction wds by means of the following formula:

$$P_{*}(\mathsf{wds} \circ \mathsf{degree}, \mathsf{satisfies} \circ \mathsf{degree}, \lambda_j^i), \tag{6}$$

The predicate $P_*(x, y)$ is true iff the condition $\mathsf{wds} * \mathsf{degree} = (\mathsf{satisfies} * \mathsf{degree}) * \mathsf{weight}$ holds. As already introduced, λ_j^i represents the relative importance of attribute j defined by user i.

Finally, the weighted degrees of satisfaction wds have to be aggregated in order to derive the overall degree of satisfaction. Analogously, this is done by the axiom

$$P_{=}(\mathsf{satisfies} \circ \mathsf{degree}, \sum a_j \circ \mathsf{wds} \circ \mathsf{degree}) \tag{7}$$

where a_j refers to the jth attribute.

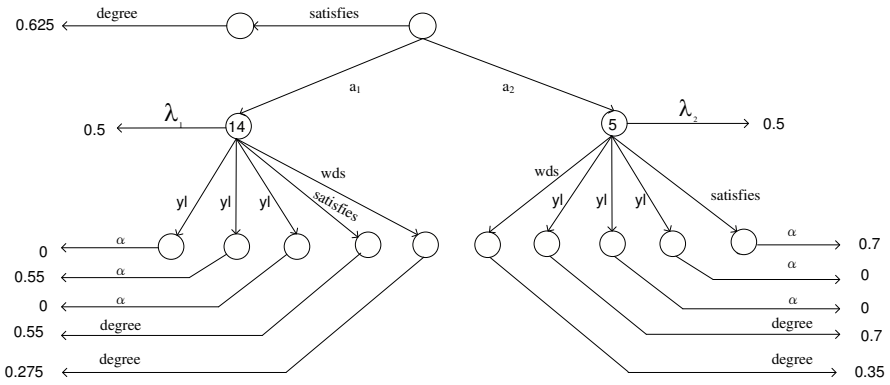

Fig. 5. Example

Example For this example the privacy policy used in section 2.1 is modified by introducing a piecewise linear function μ as follows: $\{(0, 1), (10, 0.75), (20, 0.25), (30, 0)\}$. That means the best alternative for the customer is realized when the provider does not store her private data. Consequently, the utility decreases with the number of days the private data is stored. After 10 days only 75% and after 20 days only one quarter of the overall utility remains. Beginning with a storage time of 30 days no utility can be derived from the property any more. The four points defined above result in a function containing three lines. This obviously leads to three relation instances yl in YL. For a service that stores data for 14 days the relation to all three instances of YL has to be calculated in order to derive the utility values α. Now, according to equation 6 the degree of satisfiability can be determined by aggregating the α-values. For our example, this calculation results in a degree of $0 + 0.55 + 0 = 0.55$. Analogously, the degree of satisfiability can be calculated for the attribute ResponseTime. We assume this results in a degree of 0.7 and that quality and privacy are equally important to our user ($\lambda_1 = \lambda_2 = 0.5$).

Now, the weighted degree of satisfaction wds for an attribute can be calculated by multiplying degree with the corresponding weighting factor. This results in a wds of $0.55 * 0.5 = 0.275$ for the privacy policy and $0.7 * 0.5 = 0.35$ for the quality policy. Consequently, the overall degree of satisfaction will be $0.275 + 0.35 = 0.625$.

3 Related Work

Many policy languages such as WS Policy, WS Security, EPAL, XACML, and others emerged in the Web Service community. Our work differs from these languages in that we base on a formal and extensible conceptual model. Furthermore, the WS* languages base on discrete reasoning or only vaguely define the semantics. Like our work, KAoS [12] and Rei [13] are also based on formal ontologies. In contrast to KAoS and Rei our work is currently restricted to obligations. Other modalities like permissions are not supported yet. However, both do not aim at unifying policy languages via foundational ontologies and apply a discrete reasoning approach that allows for boolean decisions only. For instance, those languages are suitable for deciding if a service is suitable according to specific policy, but make no statement about the degree of suitability. Furthermore, ontology-based policy languages often lack support for aggregation functions. This is tackled in our approach by relying on an expressive description logic ($\mathcal{ALC}(\mathcal{D}+\Sigma)$).

Moreover, in contrast to this work existing policy languages do not allow for expressing preference relations between different service configurations (e.g. between different privacy or quality levels) as well as weighting factor for the service properties. But this is necessary realizing beneficial trade-offs in a multi-attributive environment. However, some allow to assign priorities to individual policies or rules, which is not yet possible using our approach.

[14, 15] suggest to use utility functions in order to express policies and facilitate negotiation. However, they present no formal model for representing such utility information in a declarative, machine understandable, and interoperable way. But this is required to enable automatic negotiations in a distributed and heterogeneous environment. Therefore, we suggest an approach where utility information is represented by means of OWL-DL. This allows reasoning over preference information by means of standard inference engines.

Moreover, there are already existing approaches for policy based negotiation in the Semantic Web Service domain. Since deriving accurate as well as complete descriptions of Web Services is hardly manageable due to the information volume needed and the dynamic aspects that might require continuous updates of the service description, [16] introduces a contracting step where abstract service descriptions are concretized by means of individual negotiations. This procedure aims to find suitable services while keeping the descriptions simple and thus manageable. The work in this paper is complementary since we focus on the selection of the most suitable service while assuming that the set of suitable services are discovered already in the matchmaking phase before the negotiation. In [17] functional goals as well as policies are considered to find compatible services. In doing so delegation as well as trust negotiation play a crucial role, i.e. trust between two parties increases with each negotiation step. However, both approaches mentioned above allow only to derive a pure boolean statement about the compliance between policies. For selection as well as negotiation more fine-grained information about the degree of compliance might be necessary, e.g. in order to rank services or generate a counteroffer.

4 Conclusion

In this paper, we considered electronic markets as big picture and motivated the need of formal specification of policies in the allocation phase. We presented a technique for formally specifying user preferences for Web service properties and how ranking for Web services can be calculated based on such preferences. Our policy description framework is based on the foundational ontology DOLCE and thus facilitates easier integration of other policy specification languages.

In section 2.1, we have used the standard DOLCE satisfiability relation, which is in our opinion too weak. In section 2.2, we extend the satisfiability relation in a way such that one can talk about the degree of satisfiability. We have shown, how the degree of satisfiability can be calculated by a $\mathcal{ALC}(\mathcal{D}+\Sigma)$ reasoner. Note, that the description logic $\mathcal{ALC}(\mathcal{D}+\Sigma)$ is undecidable, whereas the description logic $\mathcal{ALC}(\mathcal{D})$ is decidable [18]. To be able to model the preferences with $\mathcal{ALC}(\mathcal{D})$, we only need to fix the maximum number of attributes (cf. equation (1)) and the maximum number of points in the utility function.

As discussed above, in order to enable agents to negotiate automatically without human intervention they require very detailed information about the preferences of users (e.g. utility functions for all attributes or attribute combinations, weights of the attributes). This leads to a considerable modeling effort, which obstructs the practical applicability. Thus, means have to be found to support and partly automate preference specification. Since our approach is based on utility theory we can rely on a substantial quantity of decision analysis and preference elicitation tools [19] that are already available in this context like the Analytic Hierarchy Process (AHP) [20] or a conjoint analysis.

Acknowledgements

This work was funded by the Federal Ministry of Education and Research (BMBF), the German Research Foundation (DFG), and the European Union in scope of the Internetökonomie project SESAM, the Graduate School Information Management and Market Engineering as well as the IST project SEKT.

References

1. Hurwicz, L.: The design of mechanisms for resource allocation. American Economic Review **69** (1973)
2. Lindemann, M.A., Schmid, B.F.: Elements of a reference model for electronic markets. In: HICSS '98: Proceedings of the Thirty-First Annual Hawaii International Conference on System Sciences-Volume 4. (1998)
3. Ströbel, M., Weinhard, C.: The montreal taxonomy for electronic negotiations. Group Decision and Negotiation **12** (2003) 143–164
4. Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A., Schneider, L.: The WonderWeb library of foundational ontologies. WonderWeb Deliverable D17 (2002)
5. Lamparter, S., Eberhart, A., Oberle, D.: 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. Staab, S., Studer, R.: Handbook on Ontologies. Springer Verlag, Heidelberg (2004)

7. Gangemi, A., Borgo, S., Catenacci, C., Lehmann, J.: Task taxonomies for knowledge content. Metokis deliverable d07 (2004)
8. Gangemi, A., Sagri, M.T., Tiscornia, D.: A constructive framework for legal ontologies. In Benjamins, R., Casanovas, P., Breuker, J., Gangemi, A., eds.: Law and the Semantic Web: Legal Ontologies, Methodologies, Legal Information Retrieval, and Applications. (2005)
9. Keeney, R.L., Raiffa, H.: Decisions with Multiple Objectives: Preferences and Value Tradeoffs. J. Wiley, New York (1976)
10. Russel, S., Norvig, P.: Artificial Intelligence - A Modern Approach. second edn. Prentice Hall Series in Artificiall Intelligence (2003)
11. Baader, F., Sattler, U.: Description logics with concrete domains and aggregation. In Prade, H., ed.: Proc. of the 13th European Conf. on AI (ECAI-98), John Wiley & Sons Ltd (1998)
12. Uszok, A., Bradshaw, J.M., Jeffers, R., Tate, A., Dalton, J.: Applying KAoS services to ensure policy compliance for semantic web services workflow composition and enactment. In: Int. Semantic Web Conf. (ISWC'04). (2004)
13. Kagal, L.: A Policy-Based Approach to Governing Autonomous Behavior in Distributed Environments. PhD thesis, University of Maryland, Baltimore MD 21250 (2004)
14. Kephart, J.O., Walsh, W.E.: An artificial intelligence perspective on autonomic computing policies. In: Proc. of 5th IEEE Int. Workshop on Policies for Distributed Systems and Networks. (2004)
15. Karp, A.H.: Representing utility for automated negotiation. Technical Report HPL-2003-153, HP Laboratories Palo Alto (2003)
16. Lara, R., Olmedilla, D.: Discovery and contracting of semantic web services. In: W3C Workshop on Frameworks for Semantic in Web Services, Innsbruck, Austria (2005)
17. Olmedilla, D., Lara, R., Polleres, A., Lausen, H.: Trust negotiation for semantic web services. In: 1st International Workshop on Semantic Web Services and Web Process Composition (SWSWPC). Volume 3387 of Lecture Notes in Computer Science., San Diego, CA, USA, Springer (2004) 81–95
18. Baader, F., Hanschke, P.: A schema for integrating concrete domains into concept languages. In: Proc. of the 12th Int. Joint Conf. on AI (IJCAI-91), Sydney (1991)
19. Chen, L., Pu, P.: Survey of Preference Elicitation Methods. Technical report (2004)
20. Saaty, T.L.: How to make a decision: The analytic hierarchy process. European Journal of Operational Research **48** (1990) 9–26