

Semantic-enabled Specification for Web Services Agreement

Hai Jin, Hao Wu

Cluster and Grid Computing Lab.
School of Computer Science and Technology
Huazhong University of Science and Technology, Wuhan, 430074, China
{hjin, haowu}@hust.edu.cn

Abstract: *Quality of Service (QoS)* becomes necessary for service-oriented computing. *Web Services Agreement (WSA)* aims at defining a language and a protocol for advertising the capabilities of providers, creating agreements based on creational offers, and monitoring of QoS. On the other hand, semantic web provides many burgeoning techniques that enable services intelligent and automatic on the web. Therefore, we intend to integrate semantic web techniques with WSA management to utilize its advantages. In this paper, we present some works for this study. We illustrate how to specify WSA with ontology language instead of XML schema. With this, WSA can be domain ontology to describe knowledge of service agreement, and be a unified information model for it. In addition, an agent-based runtime framework is presented for WSA management, where all agents share the WSA ontology for exchange and negotiation. This improves interoperability between these agents. The negotiation rules and policies are specially described with *Semantic Web Rule Language (SWRL)* for web service, and agents negotiate with each other under steering of such rules and policies.

Keywords: Web Services Agreement, Semantic Web, Ontology, Web Ontology Language

I. Introduction

In recent years, service-oriented computing [1] attracts more attentions, and *Quality of Service (QoS)* becomes necessary for such distributed environments. *Service Consumers (SCs)* like to have guarantees related to services. It normally depends on the resource providing capacity of *Service Providers (SPs)* that whether they can offer such guarantees. Therefore, QoS and other guarantees that depend on actual resource usage cannot simply be advertised as an invariant property of a service and then bound by a SC. Instead, the SC must request state-dependent guarantees to the provider, resulting in an agreement on the service and the associated guarantees. Additionally, the guarantees on service quality must be monitored and failure to meet these guarantees must be notified to consumers.

The objective of the *WS-Agreement (WSA)* specification [2] is to define a language and a protocol for advertising the capabilities of providers, creating agreements based on creational offers, and monitoring QoS. An agreement between a SC and a SP specifies one or more service level objectives both as expressions of requirements of the service consumer and assurances of the provider on the availability of resources and/or on service qualities. To obtain the quality assurance, the SC or its behalf must establish a service agreement with the SP, or its agent. The *WS-Agreement* just provides a

schema for defining overall structure for an agreement document. However, the specification of *WS-Agreement* is based on XML syntax and schema.

Semantic web [3] has received more attentions, because it provides many burgeoning techniques to enable intelligence and automation on the web. *Web Ontology Language (OWL)* [4] is a standard outcome in such research area. OWL is more expressive than XML with rich semantic and precisely, it provides formalized logical expression and supports logic inference on existing knowledge. OWL facilitates greater machine interpretability of web content than XML [5], *RDF* [6], and *RDF Schema (RDFS)* [6]. For this reason, we intend to integrate semantic web techniques with WSA management to exert its advantages, such as inference and intelligence. We manage to express *WS-Agreement* in OWL, so as to obtain a more powerful specification language. Grounding on this, we also introduce an agent-based runtime framework for service agreement management.

The rest of paper is organized as follows. Background knowledge and the state-of-the-art works on service agreement are reviewed and discussed in section II. We explain *WS-Agreement* structure and compare XML with OWL. Section III presents the approach on how to modeling and expressing *WS-Agreement* in ontology, and introduces some semantic-based methods to specify terms in an agreement. In section IV, we present an agent-based runtime framework for service agreement management. Also potential advantages with this approach are presented. Finally, a conclusion is given in section V.

II. Backgrounds

A. Service agreement languages

Service Level Agreement (SLA) is an important part of a contract between the client and provider of an Internet service, which describes the quality attributes that the service is required to possess. There have been several reports in this research area [7]-[12].

Web Service Offerings Language (WSOL) [7] is an extension based on *Web Service Definition Language (WSDL)*, which provides type systems for SLAs and allows the same SLA to be described in abstract and instantiated with specific values. This provides more guidance to developers of new SLAs. *WSOL* provides additional reuse facilities, including template instantiation and reuse of definitions. *Web Service Level Agreement (WSLA)* [8] is similar to *Web Services Management Language (WSML)* [9]; they both are XML-based web services SLA language. *WSLA* provides the

ability to create new metrics defined as functions over existing metrics. This is useful to formalize requirements expressed in terms of multiple QoS characteristics. The latter describes a theoretic web services QoS parameter specification model for SLA.

SLAng [10] is another XML language for defining SLA. The authors define the semantics of SLAng precisely by modeling the syntax of the language in UML. The presence of SLAng elements imposes behavioral constraints on service elements, and the precise definition of these constraints using OCL constitutes the semantic description of the language.

There are also other works to provide different modeling approaches to formalize a service agreement [11][12].

These works suggest some useful methods and tools to facilitate the service agreement specification and management. But there are still many issues needed further studied on service agreement. We summarize several sticking points as followings:

1. Lack of unified service agreement information model [13] (i.e. agreement template) for negotiation and cooperation between different entities.
2. Meanings inside service agreement are ambiguous. Different organizations define different service agreement parameters and QoS metrics; even in the same organization, different names for the same concept imply variable meanings for humans. This brings misunderstanding for service agreement constraints and contents, and goes against communication and cooperation.
3. It is hard to reuse the service agreement template and advance the service agreement development. There are some works about template for commercial purpose.
4. To enable automatically management based on service agreement information model. For example, designing automatic negotiation mechanism based on service contract model [14].

WS-Agreement is a standardization effort being conducted in GGF to solve standard and unified criterion of service agreement. We use WS-Agreement as base and develop some methods to represent service agreement with semantic web techniques.

B. XML and ontology representation languages

XML is well known in the Internet community as the basis for the rapidly growing number of software development activities. It is designed for markup in documents of arbitrary structure, as opposed to HTML, which was designed for hypertext documents with fixed structures. XML Schema defines a standard grammar to specify allowable combinations, nestings of tag names, attribute names, and so on [5]. The *Resource Description Framework* (RDF) [6] is a W3C recommendation designed to standardize the definition and application of metadata descriptions of web-based resources. RDF provides basic object-attribute-value data model for metadata, and these data models can be represented in XML syntax. Moreover, RDF is equally well suited to represent data as XML.

RDFS [6] is developed to describe properties and classes of RDF resources, with a semantic for creating hierarchies of such objects and classes and thus providing the means for generalization. RDFS is considered to be an ontology language. However, to implement the semantic web, RDFS is

not quite optimal as it lacks the features necessary to describe resources in sufficient detail. RDFS is suitable for providing the means for an ontology that characterizes some environment, no matter how abstract. RDFS alone, however, suffers from its dependence on domain-specific and case-specific details. RDFS suffers from an expressive inadequacy and it lacks a number of important relations between classes such as equivalence and disjointness, as well as cardinality and characteristics of properties.

To solve the RDFS problems, OWL is recommended by W3C for standardization. It is the combination of two originally independent ontology language DAML (*DARPA Agent Markup Language*) and OIL (*Ontology Inference Layer*), and is layered on top of RDFS to use its syntax to express ontological primitives such as Class, Relation, and Subclass. In addition OWL adds a much richer set of its own primitives, such as transitivity, cardinality, and disjunction. Also, it adds characteristics of properties like symmetry, richer typing of properties (e.g. *nonNegativeInteger*), and enumerated classes [4]. As a result, OWL has more facilities to express meaning and semantics than XML and RDF(S). Thus OWL goes beyond these languages in its ability to represent machine interpretable content on the web.

Compared with XML, OWL makes computer process information with cognitive ability of human-like (OWL uses description logic as its reasoning foundation). For the same information encoded with XML, although conforming to the XML specification, different systems may response differently. Given the same data, any system that conforms to the OWL specification will generate the same new information and conclusions, solely based on the OWL standard. A set of OWL statements in conjunction with the OWL specification, enables the conclusion of another OWL statement, whereas a set of XML statements, in conjunction with the XML specification, does not allow the conclusion of any other XML statements. To employ XML to generate new data, more assistant works are needed in XML procedural code, which is contrast to OWL, in which the knowledge is explicitly stated in OWL statements.

Another distinct difference is the ability of supporting inheritance. XML supports sub types which are restrictions of extensions on a type, but there are no classes. Consequently, there is no true notion of inheritance. Suppose that the information has inheritance relations, a data type *A* is an aggregation of two other data types *B* and *C* which comprises of subclasses *B1* and *B2*, *C1* and *C2*, respectively. If this information is encoded in XML, it needs application logic that iteratively checks all possible combinations of *B* and *C* to satisfy a query. If the same information is encoded in OWL, we only need to query the existence of *A*.

In conclusion, OWL is more suitable for purposes concluded as below:

1. Formalize a domain by defining classes and properties of those classes;
2. Define individuals and assert properties about them;
3. Reason about these classes and individuals to the degree permitted by the formal semantics of the OWL language.

C. WS-Agreements structure

Web Service Agreement is conceptually composed of several distinct parts [2]. It includes information on the agreement parties and references to prior agreements referred as

agreement context, one or more discipline specific service definition terms, one or more guarantee terms specifying service level objectives, and business values associated with these objectives. They are listed below in Figure 1.

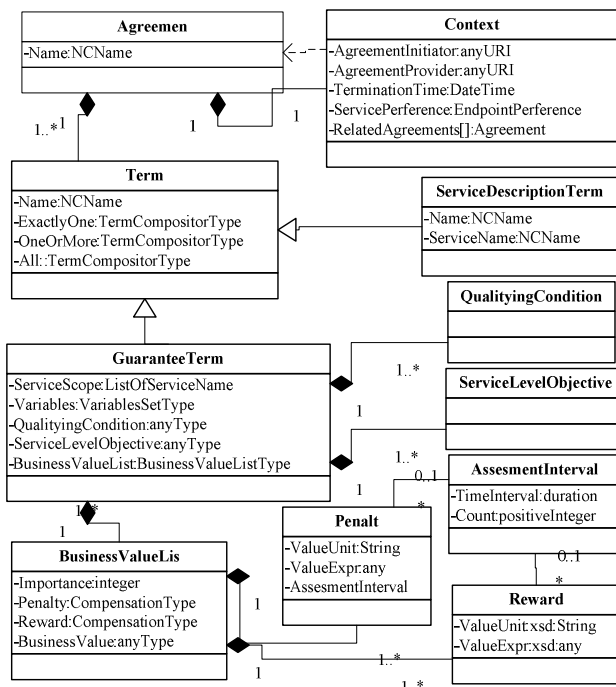


Figure 1. Main parts of WS-Agreement structure

1) Name

It describes the name of an agreement.

2) Context

The context after the name contains the meta-data describing the agreement. It names the participants in the agreement, the agreement's lifetime and links to other agreements related to this agreement.

3) Terms

The terms of an agreement are wrapped by a *wsag:Terms* term compositor. There are two types of terms: service description terms and guarantee terms. The service description terms provide information needed to instantiate or otherwise identify a service to which this agreement pertains. The guarantee terms specify the service levels that the parties are agreeing to. Management systems may use the guarantee terms to monitor the service and enforce the agreement.

The specification defines schema for service description and agreement terms as abstract types that must be extended for specific usage domain.

III. Using OWL to Express WS-Agreement

We aim at utilizing OWL as the description language to express WSA, so as to obtain more powerful and semantic-rich normal description of the WS-Agreement. The draft of the WS-Agreement specification is a basement for next step, and its schema is considered as the base vocabulary references of aftertime ontology. The implementation consists of two different translations; in the first case, we map the WS-Agreement schema to OWL ontology, while in the second case, terms grammar logic is substituted with OWL class

expression. Besides, some semantic-based methods are introduced to express items in an agreement.

A. Translating WS-Agreement schema to OWL ontology

In this section, we concern the XML Schema to OWL mapping. In section II, the differences between XML and OWL have been discussed. However, the two languages share the goal of defining common vocabularies and structures to support electronic exchange of information [15]. The goal is to design WS-Agreement ontology based on its XML Schema, and thus the mapping approach capitalizes the similarities between these two languages.

In the following, we give an overview of fundamental steps of this mapping procedure.

1) Converting main concepts

This procedure focuses on how to convert XML Schema to OWL classes or properties, including two different processes. One part, each XML Schema *complexType* is mapped to an *owl:Class*. Model group definitions and attribute group definitions are also *complexType*, which contain elements with respective to attribute declarations. The other part, elements of *simpleType* and all attributes are mapped to an *owl:DatatypeProperty*. Elements of *complexType* are mapped to an *owl:ObjectProperty*. The schema root element is mapped to an OWL class with the name of "targetNamespace+#Schema". Global element and attribute declarations are mapped similarly to local ones. Fig. 2 shows main classes translated from the WSA Schema.

2) Converting specialization relationships

XML Schema possesses two important specialization mechanisms including inheritance by restriction and inheritance by extension, both of which provide corresponding type derivation constructs. We use *rdfs:subClassOf* which is the only concept inheritance mechanism in OWL to represent this semantic. For example, in WS-Agreement schema, *GuaranteeTermType* and *ServiceDescriptionTermType* are both derived from complex type "TermType" through XSD:extension constructor. With our approach, two opposite OWL classes *GuaranteeTermType* and *ServiceDescriptionTermType* are created, and so as the subclass of OWL class *TermType*.

3) Converting type and cardinality

In XML Schema, there is also quantity constraint related to a local element or a local attribute. Because these definitions have a local scope, we map them to the intersection of two property restrictions: one restricting the type with *owl:allValuesFrom*, the other restricting the cardinality with *owl:minCardinality*, *owl:maxCardinality*, or cardinality. In WS-Agreement schema, *maxCardinality* of elements or attribute mostly is normally "unbounded", which means that the corresponding elements can iterate arbitrary times. After mapping to *owl:maxCardinality*, we set its default value at 99 (for most cases, this number is enough for an actual agreement). As for *minCardinality*, the default value is usually zero, so we let the corresponding *owl:minCardinality* zero.

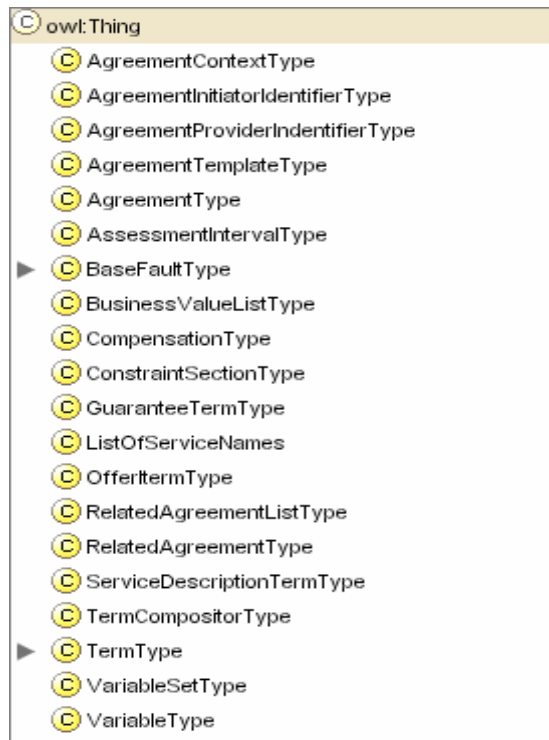


Figure 2. WS-Agreement ontology classes from corresponding XML Schema

4) Converting compositors of XML Schema

In WS-Agreement schema, three compositors are offered to combine elements, sequence, all and choice. The compositors can be represented in OWL boolean expressions, such as *owl:intersectionOf*, *owl:unionOf* and *owl:complementOf*. Sequence and all can share the same semantic replacer with *owl:intersectionOf*, that is to say, they are both conjunctions, which can be modeled in the *owl:intersectionOf* constructor. The choice compositor means an exclusive-OR, and a verbose boolean expression is needed to express its semantics. (Note that, we can skip the disposal for the compositors; as far as an agreement document is created with restrict conventional writing style.)

Due to the fact that the schema definition of WS-Agreement only adopts parts of XML language components, other disposal of language components of XML is out of the scope in this work, but there are some useful suggestions in [15] and [16].

B. Expressing terms description structure with OWL class expression

As mentioned before, the approach to create ontology consists of two important aspects. The second part translates the terms constructs of WS-Agreement from the normal expressions form to OWL class constructs, so as to grasp the semantic expression about the logic structure. The terms of an agreement are wrapped by a *wsag:Terms* term compositor. A normal form of term expression is straightforward XML Infoset representation of a *Term* statement of an agreement, enumerating each of its alternative *Service Description Terms* or *Guarantee Terms*. The form of a term expression is as follows:

```
<wsag:Terms>
<wsag:All>wsag:TermCompositorType</wsag:All> |
<wsag:OneOrMore>wsag:TermCompositorType
</wsag:OneOrMore> |
<wsag:ExactlyOne> wsag:TermCompositorType
```

```
</wsag:ExactlyOne> |
{
<wsag:ServiceDescriptionTerm>
wsag:ServiceDescriptionTermType
</wsag:ServiceDescriptionTerm> |
<wsag:GuaranteeTerm> wsag:GuaranteeTermType
</wsag:GuaranteeTerm>
} *
</wsag:Terms>
```

There are three kinds logic grouping operator in terms structure, including *wsag:All*, *wsag:OneOrMore*, and *wsag:ExactlyOne*. *wsag:All* means that all of the term assertions enclosed by this operator have to satisfy the order. Thus, it is a logical conjunction which can be represented as an intersection of OWL classes (*owl:intersectionOf*). Each of the members of the intersection is a service description term or a guarantee term, and the resulting class expression is a custom-made term class that expresses the same semantics as a term compositor one. *wsag:OneOrMore* construct captures the meaning that at least one of the alternatives in the term assertions is supported by SC or SP, and can be mapped using *owl:unionOf*.

wsag:ExactlyOne means exclusive-OR, namely, only one, not more, of the alternatives should be supported in order for the provider to support the terms. The expression of *wsag:ExactlyOne* can be substituted with: *intersectionOf (complementOf(intersectionOf(A,B)),unionOf(A,B))*, here *A* and *B* indicate items.

Using OWL class expression to represent the semantic of terms structure, we can develop our own customized agreement terms directly based on web ontology language mechanism. For example, a term statement in an access control environment is needed to assert service provider support fast response guarantee on a service, meanwhile the service provider must also offer security guarantee of access verification. This semantic implies that those two guarantees can be represented as two guarantee statements in a term alternative. We use the *owl:intersectionOf* operator to model and express the same semantic function, as shown below:

```
<owl:Class
rdf:about="http://grid.hust.edu.cn/semantic/wsa-extension.o
wl#SecureResponseGuarantee">
<owl:equivalentClass>
<owl:Class>
<owl:intersectionOf rdf:parseType="Collection">
<owl:Class rdf:about="http://grid.hust.edu.cn/
semantic/wsa-extension.owl#SecureGuarantee"/>
<owl:Class rdf:about="http://grid.hust.edu.cn/
semantic/wsa-extension.owl #ResponseGuarantee"/>
</owl:intersectionOf>
</owl:Class>
</owl:equivalentClass>
<rdfs:subClassOf>
<owl:Class rdf:about="http://grid.hust.edu.cn/
semantic/wsa.owl#GuaranteeTermType"/>
</rdfs:subClassOf>
</owl:Class>
```

The translations of other terms structures are performed in the same way as discussed above. For simplicity, they are not discussed in detail here, please refer to [17].

C. Expressing Service Level Objective semantics with rules

Service Level Objective (SLO) indicates that scale target should obey an agreement. When the SLO is dissatisfied and

failed in, the corresponding action should be taken. One original goal of WS-Agreement is to use condition specification language to define SLO and negotiation constraints. Rule-based manner is employed to express this semantic in this study. For example, to express “availability of a web service is larger than 95%”, we can adopt the rule denotation method of formalized logic. To enable rules in WS-Agreement files with a semantic and a formal expression, *Semantic Web Rule Language* (SWRL) [18] is selected as the rule language. SWRL is suitable for rule system, conditions expression, and format translation. Based on SWRL, the service level specification of web service can be described with the following formulas.

$$\begin{aligned} & \text{Agreement} (?x) \wedge \text{ResponseTime} (?y) \wedge \text{Millisecond} (?50) \wedge \\ & \text{hasSLO} (?x, ?y) \wedge \text{greaterThan} (?y, ?50) \\ \Rightarrow & \text{violationOn} (?x, ?y) \end{aligned} \quad (1)$$

$$\begin{aligned} & \text{ViolationAction} (?z) \wedge \text{violationOn} (?x, ?y) \wedge \\ & \text{actionOn} (?z, ?y) \\ \Rightarrow & \text{StartupViolationAction} (?z) \end{aligned} \quad (2)$$

Service level clauses describe violation action as well as QoS target. When an actor of an agreement audits the WSA files, it compares the QoS target with the gathering value of Service Level Objective and evaluates whether the QoS target is achieved. Once the violation happens, the notification is delivered to both sides of the agreement, and penalty action will be logged or taken. Of course, reward action plays the same procedures as penalty. One example is shown as the above expressions (It is a relatively informal “human readable” syntax in which a rule has the form *antecedent* \Rightarrow *consequent*, where both *antecedent* and *consequent* are conjunctions of atoms written $a_1 \wedge \dots \wedge a_n$. Variables are indicated using the standard convention of prefixing them with a question mark. For example, *ResponseTime* (?y) means y is an instance of SLO *ResponseTime*). The WSA has a rule about SLO response time which should be less than 50 milliseconds. However, when the delay exceeds 50 milliseconds, the breach of guarantee for response time will happen (expressed by rule1) and the corresponding penalty action should be taken (expressed by rule2).

D. Expressing QoS metrics

QoS constraints describe properties such as performance, reliability, and availability. They check whether the monitored QoS metrics are within specified limits. The exact definition and measuring process for each metric must be well-defined to give service participants a common understanding. Some ontology about QoS has been reported in [19][20]. In [19], a simple ontology is suggested to enhance the QoS model of the service profile, which provides semantic description of web service; whereas in [20] an ontology-based framework for dynamic web services selection is proposed. However, they do not address the fact that some QoS metrics, such as response time, depend on workload intensity level, which means that a single value is not appropriate. The web service cost is often related to its quality. Faster, reliable, secure services will be more expensive, but there could also be penalties associated without meeting certain QoS goals or *service-level agreements* (SLAs). Therefore the measurement process for each QoS metric should consider [21]:

1. What to measure (aggregates or percentiles);
2. How to measure (frequency, intervals, and tools);

3. Who does the measuring (service providers or external independent agents);
4. Where the measurements are taken (web service access point, network edge, or at the consumer).

In general, QoS parameters can be classified into two categories, aggregative and single. The aggregative parameters can be derived from single parameters. For instance, the reliability of one atomic service is R_i , the aggregative reliability of service flow can be expressed as

$$\prod_{i=0}^n R_i$$

Thus the metric includes atomic metric, the most

important metric which can be only obtained by single measurement, single metric and aggregative metric. Fig.3 shows the sketch map of QoS metric ontology. QoS ontology holds its own namespace and be cited by other XML-based ontology languages. This provides a domain knowledge model for QoS metric, but for specific requirement, other QoS metrics still need to be introduced.

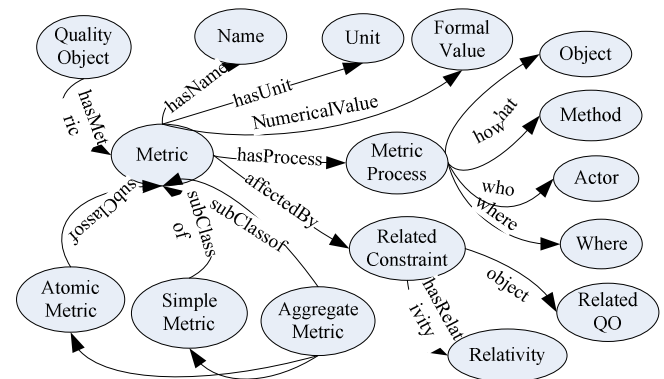


Figure 3. QoS metric ontology

IV. Integrating Services Agreement with Agent-based Runtime Framework

A software agent is a computer program that acts autonomously on behalf of a person or an organization. Agents are often characterized by the some properties, such as proactive, adaptive, autonomous, intelligent, and mobile. Nowadays, agent has been widely applied in automatic management area, and it also acts as a key role to promote intelligence and automation in semantic web environment. There are also suggestions [22][23] to apply agents to service management. Here we propose integrating web service agreement with enhanced agent-based runtime framework.

A. Rule-based automatic service contract negotiation

In order to ensure optimal service provision arrangement, all aspects of the SLA must be negotiated between the provider and the customer. Negotiation can be used for configuration setup, information transfer, new service deployment, or the usage of some physical resources such as bandwidth, CPU, and memory. The parameters involved include quality requirements, performance level, price, payment conditions, etc. The decision criteria relate to user preferences, for example, towards the fastest network access, or the closest, the most reliable, the cheapest, etc. The user could also prefer to combine service from multiple available providers in order to increase robustness.

However, the existing negotiation is mostly static, for dynamic requirement the automatic negotiation [14][24] is proposed. Automatic negotiation mimics human negotiation processes to reach agreements on one or more issues. It is an active research topic in the field of multi-agent systems, and has been applied to several areas including telecommunication and computer networks. The use of *Agent Communication Languages* (ACL) [25] enables rich and flexible interactions, which can be made interoperable through standardized specifications provided by the FIPA consortium [26]. There are also some works on how to bridge FIPA with ontology language [27]. Therefore, based on these works, we can use web ontology language and tools to specify and publish the underlying common ontology as a content language within the FIPA ACL messages.

The negotiation is a complex process which reflects human's behavior. The automatic negotiation includes two aspects: negotiation protocol and negotiation strategies. The former concept regulates the bargain rules among participants of negotiation. Negotiation strategies are dynamic, different strategies designed by person resulting in different effects. According to [14], automatic negotiation can induce into three research mode, game theoretic models, heuristic approaches and argumentation-based approaches. If strategy of automatic negotiation platform encoded by program code (such as C/C++, Java), the predefined and concrete negotiation strategies obviously are not appropriate for negotiation personnel. If we treat the negotiation strategies as knowledge statement, rule language is used to describe this knowledge, and then do logic reasoning by the inference engine. Through this ways, bargain becomes manageable and it is easier to develop more complicated strategies to satisfy the demand of negotiation personnel.

Furthermore, in order to make the negotiation agent fit comprehensive negotiation models, take part in variable automatic negotiation process, and to extend adaptability of agent, there must confront communication issues between heterogeneous and variable soft agents. We can establish the common communication method for ontology; meanwhile, ontology supports reasoning ability and helps soft agents possess more enriched knowledge reasoning ability. In general, negotiation strategy should possess complicated logic description ability. SWRL has the ability of utilizing rule language to express OWL, and can highly tie in OWL ontology. Especially, with its "Built-ins" logic constructor, SWRL is easy to build more complicated logic description.

We use OWL to express the topic for negotiation, and it can eliminate the semantic ambiguity of negotiation process. SWRL is selected to describe negotiation strategies except to express SLO semantic. Like the Service Agreement Template provides different service contracts prototype for special applications, we build negotiation strategy rules templates for different scenarios so that negotiation parties can obey coherent and stretch template to implement concrete bargain policies. Due to the complicity, the concrete bargaining policies are not discussed here.

The rule-based automatic service agreement negotiation can be pictured as follows. Suppose the negotiation processed between SC and SP, SP firstly sends its bargain information (generally are SLOs and their marked value) to SC, the SC receives and takes these information as the fact. SC uses the fact and reason with this fact combined with the private rules knowledge base, produces the dicker information according to

the rule-based deducing consequents. The dicker information will be returned to SP. Thus one round of negotiation is completed, this step will repeat until the bargain on.

When the process of negotiation completes, it should have a successful agreement. If there are requirements to modify the agreement, re-negotiation is needed.

Fig. 4 shows an agent-based supporting framework for WS Agreement, which indicates a simplified service supply model with a single service customer, one abstract service provider, which is composed of one CSP and two ISPs.

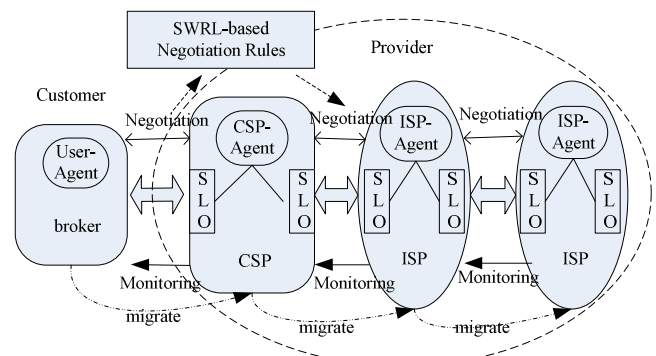


Figure 4. Agent-based WSA supporting framework

As shown in Fig. 4, there are agents associated with all participants in the provisional chain. The agent is mobile, and can be released and migrated from one participant to another. These agents perform management functions of service agreement, such as negotiation, monitoring and evaluation. The agent mainly focuses on *Service Level Objective* (SLO) item in an agreement while negotiation. All agents share the WSA ontology which offers the common vocabularies for exchange and negotiation in the service agreement domain. This improves interoperability among these agents. The negotiation rules and policies are described with SWRL for specific web service, and the agents negotiate with each other under steering of such rules and policies. When performing monitoring, the agent connects with the monitoring devices, which could be agents or commercial monitoring tools embedded in service providers to achieve SLO related service parameters such as delay and bandwidth.

After service discovery, the SC contacts with the SP. Here the *content service provider* (CSP) provides firsthand web services for the SC, whereas the ISP offers supporting service for the upper web service. The customer releases its *User-Agent* (UA) which migrates to the CSP and negotiates with *CSP-Agent* (CA), then bargains on the agreement between end customer and the CSP. The CSP performs the same procedure. It sends its CA to the *ISP1* and negotiates with *ISP1's agent* (*ISP-Agent*, IA). *ISP1* also performs the same procedure with *ISP2*. When all the negotiations are accomplished, the provisioning chain is created. Then, all agents have migrated to the different platforms with the connection to local monitoring devices, and perform monitoring during service provisioning with the QoS constraints in the SLA. After the service and the monitoring being finished, the agents will be deleted.

B. Potential impacts

There are several advantages over the works mentioned in section II.

First, the service agreement is expressed in OWL instead of XML or other languages. For example, we use OWL class

expression to capture the “Terms” statement semantic. It specifies service agreement with rich semantics, whereas other XML-based languages lack this talent.

Second, it boosts domain knowledge and code reusability. The works mentioned in section II provide some reuse facilities, such as template instantiation and reuse of definitions. These reuse facilities are reuse of data structure, whereas WSA ontology provides the common knowledge of SLA domain and can achieve reuse of knowledge level rather than data structure level. For example, we can abstract the generalization relationships (may be taxonomy) between SLAs and build a family of SLA types, and then adopt ontology to describe the abstract relationships and form a SLA type’s knowledge repository for reuse. Several well-defined industry standard taxonomies have existed, such as the *North American Industry Classification System* (NAICS, <http://www.census.gov/epcd/www/naics.html>) and UNSPSC (<http://eccma.org/unspsc/browse/>). They may facilitate the creating works of template base.

Third, it improves interoperability. Suppose two service providers offer similar services, and they define different names and metrics for the same service. This is a typical interoperable issue which is rather troublesome for XML, whereas it can be easily settled with ontology mapping mechanism, by which we set two names equivalent and present the conversion table between two measurements.

Finally, the semantic enhanced agent-based supporting framework is proposed for WS-agreement management, and it can provide better automation and intelligence. The agent performs intelligent inference based on rules supported by semantic web techniques. This facilitates automation of the negotiation, the monitoring and the evaluation of service agreement. The well-established reasoning tool is also a great help to check the consistency between agreements.

Nowadays, the ontology (OWL-S/WSMO [28]) and semantic web technique have been applied for web service automatic discovery and composition. As compared with OWL-S, our work focuses on providing a contract to restrict carrying out of the service, whereas the former provides functional and non-functional semantic description of service so as to discover and composite services automatically. Furthermore, in OWL-S, the QoS model is incomplete and the semantic and functions of SLA remain questionable, our work can be seen as supplements.

V. Conclusion

We have proposed an approach to express a web service agreement in OWL and described how to create ontology based on the WS-Agreement Schema. Some novel approaches are employed to represent the formalism aspects of WS-Agreement. We also propose an agent-based supporting framework for agreement management. To the best of our knowledge, up to the present, there is still no complete work which specifies service agreement in ontology language. Because the semantic web techniques combining with Web Service Agreement remains questionable, our work is an attempt for this.

Acknowledgment

This work is supported by National Basic 973 Research

Program of China under grant No.2003CB317003. Furthermore, we are grateful to other members in our team for their creative advices.

References

- [1] M. P. Papazoglou and D. Georgakopoulos. “Service-Oriented Computing”, *Communication of the ACM*, 46(10), pp.25-28, 2003.
- [2] H. Ludwig, A. Dan, and R. Kearney. “Cremona: An Architecture and Library for Creation and Monitoring of WS-Agreements”. In *Proceedings of the Second International Conference on Service Oriented Computing (ICSOC '04)*, pp.65-74, 2004.
- [3] H. Takeda. “Semantic Web: a Road to the Knowledge Infrastructure on the Internet”, *New Generation Computing*, 22(4), pp.395-413, 2004.
- [4] I. Horrocks, P. F. Patel-Schneider, F. van Harmelen. “From SHIQ and RDF to OWL: The Making of a Web Ontology Language”, *Journal of Web Semantics*, 1(1), pp.7-26, 2003.
- [5] R. Khare and A. Rifkin. “XML: a Door to Automated Web Applications”, *IEEE Internet Computing*, 1(4), pp.78- 87, 1997.
- [6] S. Decker, S. Melnik, F. van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann, and I. Horrocks. “The Semantic Web: the Roles of XML and RDF”, *IEEE Internet Computing*, 4(5), pp.63-73, 2000.
- [7] V. Tosic, B. Pagurek, and K. Patel. “WSOL-a Language for the Formal Specification of Classes of Service for Web Services”. In *Proceedings of the International Conference on Web Services*, pp.375-381, 2003.
- [8] A. Keller and H. Ludwig. “The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services”, *Journal of Network and Systems Management*, 11(1), pp.57-81, 2003.
- [9] A. Sahai, V. Machiraju, M. Sayal, A. van Moorsel, and F. Casati. “Automated SLA Monitoring for Web Services”. In *Proceedings of 13th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*, pp.28-41, 2002.
- [10] J. Skene, D. D. Lamanna, and W. Emmerich. “Precise Service Level Agreements”. In *Proceedings of 26th International Conference on Software Engineering*, pp.179-188, 2004.
- [11] M. Debusmann, M. Schmidt, M. Schmid, and R. Kroeger. “Unified Service Level Monitoring Using CIM”. In *Proceedings of 7th IEEE International Enterprise Distributed Object Computing Conference*, pp.76-85, 2003.
- [12] M. Debusmann, R. Kroger, and K. Geihs. “Unifying Service Level Management Using an MDA-based Approach”. In *Proceedings of IEEE Network Operations and Management Symposium*, pp.801-814, 2004.
- [13] E. Marilly, O. Martinot, S. Betge-Brezetz, and G. Delègue. “Requirements for Service Level Agreement Management”. In *Proceedings of 3rd IEEE Workshop on IP Operations and Management*, pp.57-62, 2002.
- [14] N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge. “Automated Negotiation: Prospects, Methods and Challenges”, *International*

Journal of Group Decision and Negotiation, 10(2), pp.199-215, 2001.

- [15] M. Ferdinand, C. Zirpins, and D. Trastour. "Lifting XML Schema to OWL". In *Proceedings of 4th International Conference Web Engineering*, pp.354-358, 2004.
- [16] M. Klein, D. Fensel, F. van Harmelen, and I. Horrocks. "The Relation between Ontologies and XML Schemas", *Link oping Electronic Articles in Computer and Information Science*, 6(4), 2001.
- [17] B. Parsia, V. Kolovski, and J. Hendler. "Expressing WS Policies in OWL". In *Proceedings of Policy Management for the Web Workshop, 14th International World Wide Web Conference*, 2005.
- [18] I. Horrocks, P. F. Patel-Schneider, S. Bechhofer, and D. Tsarkov. "OWL Rules: a Proposal and Prototype Implementation", *Journal of Web Semantics*, 3(1), pp.23-40, 2005.
- [19] C. Zhou, L. T. Chia, and B. S. Lee. "DAML-QoS Ontology for Web Services". In *Proceedings of IEEE International Conference on Web Services*, pp.472-479, 2004.
- [20] E. M. Maximilien and M. P. Singh. "A Framework and Ontology for Dynamic Web Services Selection", *IEEE Internet Computing*, 8(5), pp.84-93, 2004.
- [21] D. A. Menasce. "Composing Web Service: a QoS View", *IEEE Internet Computing*, 8(6), pp.88-90, 2004.
- [22] H. Kneer, H. Stormer, B. Stiller, H. Hauschen. "An Agent-based Framework for Monitoring Service Contracts". In *Proceedings of Third International Conference on E-Commerce and Web Technologies*, pp.139-151, 2002.
- [23] F. De Turck, S. Vanhastel, P. Backx, B. Duyburgh, and P. Demeester. "Design of a Generic Architecture for Service Management and Monitoring of Service Level Agreements through Distributed Intelligent Agents". In *Proceedings of IEEE Intelligent Network Workshop*, pp.50-57, 2001.
- [24] M. Klein, P. Faratin, H. Sayama, and Y. Bar-Yam. "Protocols for Negotiating Complex Contracts", *IEEE Intelligent Systems, Special Issue on Agents and Markets*, 18(6), pp.32-38, 2003.
- [25] J. Pitt and A. Mamdani. "Some Remarks on the Semantics of FIPA's Agent Communication Language", *Autonomous Agents and Multi-Agent Systems*, 2(4), pp.333-356, 1999.
- [26] P. D. O'Brien. "FIPA-Towards a Standard for Software Agents", *BT Technology Journal*, 16(3), pp.51-59, 1998.
- [27] Y. Y. Zou, T. Finin, L. Ding, H. Chen, and R. Pan. "Using Semantic Web Technology in Multi-Agent Systems: a Case Study in the TAGA Trading Agent Environment". In *Proceedings of Fifth International Conference on Electric Commerce*, pp.95-101, 2003.
- [28] R. Lara, D. Roman, A. Polleres, and D. Fensel, "A Conceptual Comparison of WSMO and OWL-S". In *Proceedings of Web Services European Conference*, pp.254-269, 2004.

Author Biographies

Hai Jin received his B.S. degree in computer science, Master degree in computer science, and Ph.D. degree in computer engineering from Huazhong University of Science and Technology (HUST, China) in 1988, 1991, and 1994, respectively. Dr. Jin now is a professor of computer science and engineering, the director of Cluster and Grid Computing Lab, and the Dean of School of Computer Science and Technology at the HUST in China. He also is the chief scientist of the ChinaGrid Project. His research interests include computer architecture, cluster computing, grid computing, parallel and distributed computing, semantic web, peer-to-peer computing, network storage, network security, and pervasive computing.

Hao Wu received his B.S. degree in computer science from Zhengzhou University, China in 2001, and Master degree in computer science from Huazhong University of Science and Technology (HUST, China) in 2004. He is currently a Ph.D. candidate in School of Computer, HUST. His research interests include grid computing, peer-to-peer knowledge management, semantic web and parallel and distributed computing.