

A semantic based framework for supporting negotiation in Service Oriented Architectures

Marco Comuzzi
Dept. of Computing
City University London
Northampton Square, London EC1V 0HB, UK
sbbd286@soi.city.ac.uk

Kyriakos Kritikos and Pierluigi Plebani
Dip. Elettronica ed Informazione
Politecnico di Milano
Via Ponzio 34/5 - 20133 Milano - Italy
{kritikos,plebani}@elet.polimi.it

Abstract—Negotiation is required before invoking a service in order to identify how the invocation must occur in terms of functional and non-functional criteria. This process is possible when all the involved parties agree on the same negotiation protocol (e.g., bilateral negotiations). Considering a Service Oriented Architecture (SOA), this negotiation protocol cannot be predefined, but it must be selected by considering the negotiation capabilities of the involved services.

In this work, we propose a semantic-based framework for supporting the negotiation in SOA. Specifically, the framework allows to express the negotiation capabilities of service requesters and providers and proposes a mechanism for discovering the negotiation protocols that can be enacted when a negotiation is required. To improve the flexibility of the framework, the concept of delegation is introduced to deal with the situation in which a party, that is not able to support the negotiation protocol, wants to participate in a negotiation. In this case, the negotiation can be fully or partially delegated to one or more other parties that, instead, are able to support the negotiation protocol.

Keywords-QoS Negotiation; Service Oriented Architectures; Semantic Web;

I. INTRODUCTION

Service discovery is one of the fundamental activities in Service Oriented Architectures (SOA). Based on a set of functional and non-functional requirements, service discovery produces a set of available services. In the literature [1], several work deal with this issue proposing matchmaking algorithms that compare the requirements with the offerings. Different matchmakers are proposed accordingly to the different languages that can be used to describe a service: e.g., WSDL/SAWSDL, OWL-S, for the functional aspects; WS-Policy, OWL-Q [2] for the non functional aspects.

Considering the extended SOA [3], the matchmaking activity can be followed by a negotiation activity that allows to identify the Service Level Agreement (SLA) as a composition of Service Level Objectives (SLOs) (i.e service quality bounds) of the delivered service, the penalties enforced when they are violated, the measurement and evaluation process of the SLOs, the price to be payed, and other important terms.

Focusing on the negotiation, current solutions [4] start from the assumption that all the negotiating parties agree and have the capabilities to support the same negotiation protocol. Although this assumption is mandatory in a negotiation system, in SOA service requesters and service providers might not know each other in advance. This means that it is quite probable that negotiation cannot be enacted, since a common supported negotiation protocol is not easy to be found.

The goal of this work is to overcome to this limitation by proposing a semantic-based framework for supporting the negotiation in SOA. The framework assumes that all the parties specify their requirements and offers according to the same model, and a matchmaking algorithm to compare them exists [5], [6]. Our framework completes the picture by proposing a semantic-based model for allowing the participants to express their negotiation capabilities. When a negotiation is required, the framework can be used to identify which are the possible negotiation protocols by comparing the negotiation capabilities exposed by the participants. As a way to improve the flexibility of the solution accordingly to a SOA, the framework includes the concept of *negotiation by delegation*. When a negotiation protocol supported by all the participant does not exist, then we assume that a participant can fully or partially delegate the negotiation to someone else that supports other negotiation protocols that make possible to find a common protocol.

The reasoning required during the negotiation protocol matchmaking relies on an ontology obtained extending the one introduced in our previous work [7]. Such an ontology includes the description of the negotiation capabilities of the participants, automates the quality specification alignment and matchmaking processes, and assists the actual negotiation process through reasoning with rules.

The paper is structured as follows. In Section II, we clearly define the scenario in which the negotiation framework operates. Section III introduces all the elements composing the framework and gives a formal definition of them by specifying an ontology. By using an example, Section IV introduces the rules that drive the discovery of

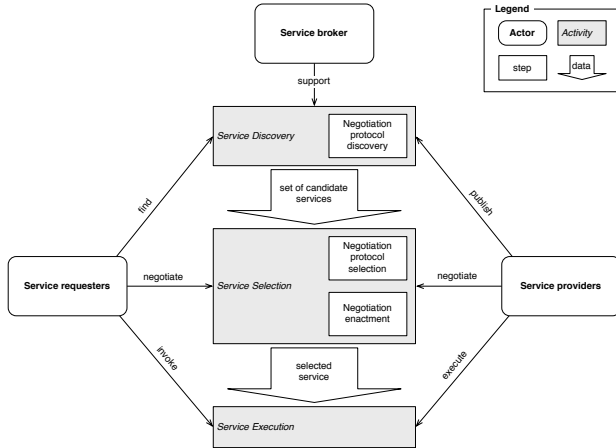


Figure 1. Negotiation-enabled SOA.

the negotiation protocols. Section V discusses some of the related work and, finally, Section VI concludes the paper and outlines possible future work.

II. SCENARIO

Before discussing in depth the negotiation framework that we are proposing in this paper, it is worth noting to draw the overall picture (see Figure 1) and to clearly identify when and how the negotiation is involved in a SOA. Here, three main players are involved: *service requesters* (or *service users*), the *service providers*, and the *service broker*. As a very first step, we assume that service providers make their service publicly available using the functionalities offered by the service broker, for instance, by means of a registry. On the other side, the service broker supports the service users during the *service discovery* activity to retrieve the published services that satisfy the user requirements. As a result a set of candidate services are retrieved and the requester is in charge of selecting the best one (i.e., *service selection* activity). Once this service has been identified, the *service execution* can commence.

This process requires that all the providers and requesters are described in terms of functional and non-functional capabilities. These capabilities are publicly available to everyone using standard documents like WSDL or WS-Policy [8] attached to the related service descriptions that we assume are already made available by the providers and requesters. These documents feed the service discovery and service selection activities in order to identify the service to be executed according to one of the several approaches now available in the literature [1].

If the capability documents include information about the negotiation protocols that a provider or a requester are able to support then the service discovery and service selection activities include three steps specifically devoted to the analysis of the negotiation protocols: *negotiation protocol*

discovery, *negotiation protocol selection*, and the *negotiation enactment*. The first step is in charge of identifying the set of negotiation protocols that are supported by all the service providers and requester involved in a given application according to the published capability documents. If there is not any negotiation protocol in common between a service provider and the requester, the negotiation cannot occur. The second and third phase belong to the service selection activity: the negotiation protocol selection is in charge of selecting the best negotiation protocol among the ones previously discovered, the negotiation enactment executes the selected negotiation protocol to define a SLA between the service provider and the service requester. If an agreement is made, then the service is executed.

Considering this scenario, the framework proposed in this paper deals with the negotiation protocol discovery. The negotiation protocol selection will be a topic of future work, whereas the negotiation enactment has been extensively covered in the literature [4].

As discussed in the literature, several protocols are available to perform a negotiation between parties. On first approximation, we distinguish between *bilateral negotiations* and *auctions* as ways to carry out a negotiation protocol. In the former case, we have only two parties involved; on the latter case, we have an auctioneer and a set of one or more participants. To make the negotiation feasible, one of the constraints requires that all the involved parties agree on the same negotiation protocol: i.e., they agree on the sequence of actions to be performed in order to conclude the negotiation process. For instance, in case of bilateral negotiation, an offer is initially generated by a party. Then, this offer can be accepted as is by the other party or a counteroffer might be generated and evaluated by the first party that behaves in the same way. The process stops when both parties agree on a given offer that satisfies both of them. In this case, in order to make possible the negotiation, the first party must be able to perform three activities: “generate the initial offer”, “generate the counteroffer”, “evaluate offer”. On the other hand, the other party must be capable for performing at least two activities: “generate the counteroffer”, “evaluate offer”. Similarly, if we consider auctions, auctioneer and participants must be able to perform particular actions considering that the participants are competing each other.

It is worth noting that we no longer talk about providers and requesters during the negotiation, but we talk about parties, participants, and auctioneers according to a given negotiation protocol. Indeed, considering for instance an auction, a requester could be auctioneer during an execution of the auction, whereas it could hold the role of participant in some other executions. Let us consider the case of a user that requires the functionality of a certain service to complete an application package that he or she would like to sell to other potential consumers. In a first phase, the user takes the role of service requester and negotiates with other

service providers in order to obtain the missing functionality for his or her application package. In a second phase, the same user takes the role of a service provider, looking for potential service requesters for its application package.

The bargain power owned by the requester and the provider is one of the most important criteria that can drive the assignment of the role during the negotiation protocol. Assuming that we are able to know this bargain power, in case of bilateral negotiation who has the main power will start the negotiation. Considering the auction, the scenario is a little bit different. After the service discovery, the requester retrieves a set of functionally equivalent services. If the requester has the main bargain power it means that the requester holds the role of the auctioneer and the services compete to make a deal with it. If the requester has less bargain power with one of retrieved service, then the situation is the opposite and this requester belongs to a set of requesters that compete for the right to use a given service.

III. NEGOTIATION FRAMEWORK

In order to better clarify our definitions of negotiation capabilities, we rely on a classification of elements in a generic negotiation that is well understood in the agent computing literature [4]. Specifically, a generic negotiation is constituted by (i) a *Negotiation Protocol*, which states the admissible actions that a participant can endorse and the valid states for a negotiation, (ii) *Negotiation Objects*, i.e. what is under negotiation, and (iii) *Decision Models*, i.e. the set of rules or reasoning models adopted by a participant to evaluate and generate offers.

The negotiation capabilities in our negotiation framework are defined in terms of the ontology that extends OWL-Q [2], a semantic quality-based service description language. In this work, the ontology is limited to the concepts required for the discovery of a *Negotiation Protocol* compatible with the negotiation capabilities defined by the potential participants. Our focus is therefore on capturing in an ontology actors, roles, and possible actions that an actor can perform or that a role may require in a given negotiation protocol as shown in Figure 2. Apart from the usual advantages of the use of an ontology, another advantage is that term matching rules can be created in cases where domain experts differently interpret the ontology's terms.

In our previous work [7], we focused on the description of *negotiation objects* and the negotiators' *decision models*, by extending OWL-Q to automate the quality specification alignment and matchmaking processes and assist the actual negotiation process through reasoning with rules.

A. Negotiation Actors and Roles

Our framework makes a clear distinction between *negotiation actors* and *negotiation roles*. The former can be *users* or *agents*. Users delegate agents (see object property *actsFor*) to take a specific role in a negotiation and act on

behalf of them. The latter can be *service providers* or *service requesters* or *brokers*. Besides expressing capabilities for participating in a negotiation protocol, service providers and requesters have the capability of generating and evaluating offers within a negotiation. Conversely, brokers only expose capabilities that do not involve the generation or evaluation of negotiation offers.

By introducing a distinction between actors and roles, our model enables a negotiation actor to take different negotiation roles in different negotiations as discussed at the end of the previous section. In the ontology model, negotiation actors are associated with roles through the object property *takesRole*.

B. Negotiation Protocols

Concerning the description of the *negotiation protocol*, the ontology model has an entity named *NegotiationProtocol* that represents all possible negotiation protocols as, for instance, bilateral negotiations, auctions, and so on. A specific negotiation protocol can be compatible with another one (i.e. it can be substituted by it) and this fact is captured with the object property *compatible*.

Negotiation Protocols and Actors are linked by the concept of *RoleSkeletons*. Role skeletons are specific roles embodied in a negotiation protocol by an actor. A negotiation protocol may have a specific set of valid role skeletons (see the object property *skeleton*) that the actors participating in it can support. At the current stage, our ontology includes only RoleSkeletons required by the bilateral negotiation and auction protocols. In a bilateral negotiation, we consider the *Initiator* and *Participant* role skeletons. The former is embodied by the actor that makes the first offer in the negotiation, whereas the latter by the second actor involved in the negotiation. Besides the *Participant*, the auction protocol involves also an *Auctioneer*, i.e. the actor in charge of collecting offers, making a decision on the auction termination, and communicating the outcome to the participants.

C. Actions

The ontology model supporting our framework relates both negotiation actors and role skeletons with a set of actions (i.e., an *ActionList*) through the object properties *ableToPerform* and *performs*, respectively.

The identification of candidate actions in our ontology takes inspiration from the large body of literature on configurable platforms for supporting generic negotiations in agent-based computing literature [9], [10]. We distinguish among actions that can be performed during the engagement of a negotiation (i.e., *engagement actions*) and actions that belong to the actual negotiation protocol (i.e., *negotiation protocol actions*). The engagement phase represents the initial setup of the negotiation, where an actor can delegate part of the actions required in a negotiation to other actors.

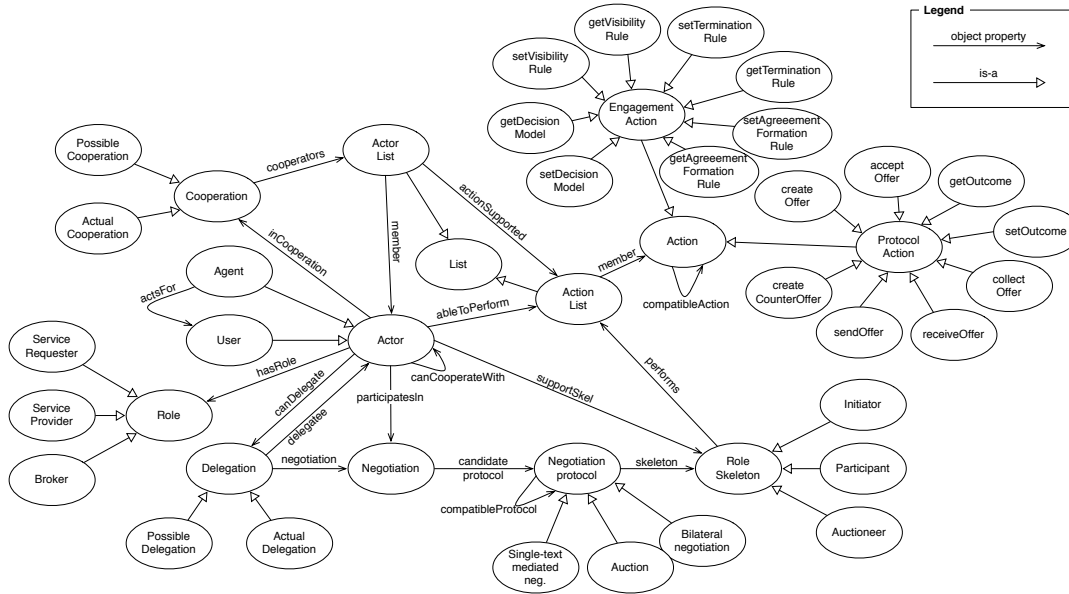


Figure 2. OWL-Q based ontology for negotiation capabilities.

Engagement Actions.

An engagement action represents a capability used by an actor to delegate some *aspects* of a negotiation to another actor. Besides the decision model, among other aspects that can be delegated, we consider the *Visibility*, *Termination*, and *Agreement Formation* rules [10].

In a multiparty negotiation, a *visibility rule* specifies the conditions under which an offer can be made visible, i.e. communicated, to the other participants of a negotiation. In an auction, for instance, an offer can be visible to other participants only when it exceeds the reservation value set by the auctioneer, which may not be known to the participants. In a generic multiparty negotiation, only the latest offer made by a participant may be visible, i.e., when an offer is submitted by agent A, all the previous offers made by A should no longer be visible to other involved agents.

A *termination rule* defines the conditions under which a negotiation terminates and, therefore, the outcome should be sent to the participants. A simple termination rule in a bilateral negotiation may be a timeout, i.e. the negotiation terminates, without an agreement, when a pre-specified deadline expires (or when an agreement has been reached before the deadline).

An *agreement formation rule* specifies the conditions under which an agreement is reached among the participants. In a bilateral negotiation, the agreement is formed when a party accepts an offer made by the counterpart. More complex rules can be defined for complex multiparty negotiation protocols. In double auctions, for instance, agreement formation rules that satisfy specific equilibrium and incentive-compatibility properties are continuously object of research studies [11].

Visibility, Termination, and Agreement formation rules, or decision models for generating offers, can be delegated by an actor (*set* actions) to another actor, which is able to receive a rule through the correspondent *get* action.

Negotiation Protocol Actions.

The set of negotiation actions considered at the current stage in our framework are the basic actions that can be used to execute a bilateral iterated negotiation protocol and a generic auction. These involve actions for creating an offer from scratch, i.e. for starting a negotiation or bidding in an auction (*createOffer*), generating a counteroffer, given an offer submitted by a counterpart (*createCounterOffer*), and accepting an offer (*acceptOffer*).

Besides these, an actor may expose an action to receive the outcome of a negotiation when it has terminated (*getOutcome*). From the broker perspective, we include actions for collecting offers from one or a set of actors (*collectOffers*) and for communicating the outcome of a negotiation when it has terminated (*setOutcome*). Finally, the actions *sendOffer* and *receiveOffer* are required for delegating part of the negotiation protocol execution to a broker. It must be noted that specific actions appearing in negotiations, such as *createOffer*, are sub-classes of the *Action* class. In this way, specific implementations of the same action will belong to the class of this action so it will be easier to perform matchmaking between them.

In our ontology model, negotiation actors publish the actions they are able to perform when involved in negotiations. Each role skeleton required by a specific negotiation protocol is associated with a set of actions as well. Such set contains the actions that a role skeleton should perform during the enactment of the protocol and, therefore, the actions that are

required by an actor supporting the role skeleton. The goal of our negotiation protocol discovery is to match the actions an actor can take with the actions of the role skeleton it wants to support in a specific negotiation protocol and, thus, to infer whether the actor can actually support the negotiation protocol or not.

The object property *supportsSkel* between the *Actor* and *RoleSkeleton* classes captures the existence of match between actors and role skeletons action lists. This property has two sub-properties, namely *fSupportsSkel* and *delegatesSkel*, which refer to two different cases of match. The former property is used to capture the fact that an Actor can fully support a role skeleton, i.e. its action lists contains the same set of actions required by the correspondent role skeleton. The latter property is used to capture the fact that a *User* has an agent acting for him that can fully support a specific role skeleton.

D. Delegation

A negotiation actor may extend its negotiation capabilities by cooperating with other actors. According to our model, this cooperation should be able to be performed in both ways and irrespectively of the negotiation protocol. So both cooperating actors should be able to perform any engagement action. This relationship between negotiation actors is captured through the symmetric object property *cooperatesWith*. An actor may need to match a role skeleton, but it may also not support all the actions required by such a role skeleton. In this case, the actor can delegate to a cooperating actor all or a subset of the actions of a role skeleton. When the whole set of actions required by a role skeleton is delegated by an actor to another one, we have the case of (full) *Delegation*. The *Delegation* class is separated into two sub-classes, i.e. *PossibleDelegation* and *ActualDelegation*, and it is related to the delegating and the delegated actors through the properties *canDelegate* and *delegatee*, respectively, and with the supported role skeleton through the property *skel*.

The *PossibleDelegation* class captures delegations that are possible to happen but their “realization” depends on three possible facts: a) a negotiation selects the negotiation protocol of the role skeleton; b) the delegator cannot support or delegate any other role skeleton; c) the delegatee does not participate in the same negotiation with the delegator and the delegator cannot fully support the role skeleton. If these three facts happen, then a possible delegation becomes an *ActualDelegation*. An *ActualDelegation* is connected with its selecting *Negotiation* with the property *negotiation*.

When an actor delegates its unsupported actions to one or more cooperating actors, then we have the case of partial delegation. This case is captured by the class *Cooperation*. This class is separated into two sub-classes in the same way as it is done for the *Delegation* class. Moreover, this class is connected with the delegating actor and all cooper-

ating actors (including the delegator) through the properties *canCooperate* and *cooperators*, respectively, and with the supported role skeleton through the property *ofSkeleton*. The *PossibleCooperation* class captures those cooperations between actors that are likely to happen in order to support a role skeleton, while the *ActualCooperation* class captures those cooperations that are going to take place with respect to a negotiation. So the *ActualCooperation* class is connected with a negotiation through the property *inNegotiation*.

Finally, our model captures all appropriate information about a negotiation through the class *Negotiation*. A *Negotiation* is connected with its participating actors through the property *participatesIn*. Moreover, a negotiation is connected with all the protocols that can be supported by its actors with the object property *candidateProtocols*. In addition, all the roles that participants have in a negotiation are captured through the object property *ofNegotiation*.

IV. SEMANTIC MODEL

The ontology in our framework aims at supporting the negotiation protocol discovery phase, i.e. finding a feasible protocol among a set of actors, given the negotiation capabilities exposed by these actors. To this end, we define specific (helper) relationships (i.e. object properties) between entities in our ontology and a set of rules to enable reasoning about the facts we want to infer. In the remainder of this section, we first discuss the relationships and rules added to support negotiation protocol discovery. We then provide a complete example illustrating the reasoning capabilities of our framework.

The first relationship added in the ontology model is *pSupportsSkel*, to describe the fact that a specific actor partially supports a specific role skeleton. Other two additions to the model are the *pSupportSkel* and *supportSkel* relationships, which capture the fact that a specific set of actors (*ActorList*) can partially or fully support a role skeleton, respectively (without considering the fact if they can cooperate or not). Another relationship added is the *actionsSupported*, to capture the fact that a set of actors can support a specific set of actions (without knowing if they can cooperate or not). Finally, we added the symmetric and transitive relationship between actors *canCooperateWith*, to capture the fact that two actors can cooperate either immediately or through other agents. For this reason, we made this new property a super-property of the *cooperatesWith* property.

The negotiation protocol discovery phase is enabled by a new set of rules added to the ontology model. These rules in First Order Logic (FOL) form can be seen in Figure 3.

Rules (1)–(5) reveal the logic for inferring the candidate protocols that can be used in a specific negotiation. Rule (1) states that an actor fully supports a role skeleton when he is able to perform all the actions that the role skeleton should perform. Rule (2) states that an actor delegates a role skeleton when he has an agent acting for him and this agent

1. $fSupportsSkel(A, S) \leftarrow ableToPerform(A, L_1) \wedge performs(S, L_2) \wedge subList(L_2, L_1)$
2. $delegatesSkel(A, S) \leftarrow actsFor(A_2, A) \wedge fSupportsSkel(A_2, S)$
3. $PossibleDelegation(D) \wedge canDelegate(A, D) \wedge delegatee(D, A_2) \wedge skel(D, S) \leftarrow$
 $\leftarrow cooperatesWith(A, A_2) \wedge supportsSkel(A_2, S)$
4. $PossibleCooperation(C) \wedge inCooperation(A, C) \wedge cooperators(C, L) \wedge ofSkeleton(C, S) \leftarrow$
 $\leftarrow supportSkel(L, S) \wedge contains(L, A) \wedge \forall X(member(L, X) \wedge differentFrom(X, A) \rightarrow$
 $\rightarrow canCooperateWith(A, X))$
5. $candidateProtocol(N, P) \leftarrow newList(LS) \wedge \forall R(ofNegotiation(R, N) \wedge \exists S(skeleton(P, S) \wedge$
 $\neg member(LS, S) \wedge \forall X(participatesIn(X, N) \wedge takesRole(X, R) \wedge (supports(X, S) \vee ($
 $canDelegate(X, D) \wedge delegatee(D, X_2) \wedge skel(D, S) \wedge \neg participatesIn(X_2, N))) \vee ($
 $inCooperation(X, C) \wedge cooperators(C, L) \wedge ofSkeleton(C, S) \wedge \neg \exists X_2(member(L, X) \wedge$
 $differentFrom(X_2, X) \wedge participatesIn(X_2, N)))) \wedge listAdd(LS, S)))$
 $\wedge \forall S(skeleton(P, S) \wedge member(LS, S))$
6. $pSupportsSkel(A, S) \leftarrow ableToPerform(A, L_1) \wedge performs(S, L_2) \wedge$
 $\wedge listIntersection(L_3, L_1, L_2) \wedge strictlySubList(L_3, L_2)$
7. $pSupportSkel(AL, S) \wedge actionsSupported(AL, L) \leftarrow pSupportsSkel(A, S) \wedge$
 $\wedge ableToPerform(A, L) \wedge listAdd(AL, A)$
8. $pSupportSkel(AL_2, S) \wedge actionsSupported(AL_2, L_4) \leftarrow pSupportSkel(AL_1, S) \wedge$
 $\wedge pSupportsSkel(A, S) \wedge ableToPerform(A, L_1) \wedge actionsSupported(AL_1, L_3) \wedge$
 $\wedge listConcatenation(L_4, L_1, L_3) \wedge performs(S, L_2) \wedge listIntersection(L_5, L_4, L_2) \wedge$
 $\wedge strictlySubList(L_5, L_2) \wedge newList(AL_2, AL_1, A)$
9. $supportSkel(AL_2, S) \wedge actionsSupported(AL_2, L_4) \leftarrow pSupportSkel(AL_1, S) \wedge$
 $\wedge (A, S) \wedge ableToPerform(A, L_1) \wedge actionsSupported(AL_1, L_3) \wedge$
 $\wedge listConcatenation(L_4, L_1, L_3) \wedge performs(S, L_2) \wedge subList(L_2, L_4) \wedge newList(AL_2, AL_1, A)$

Figure 3. Rules for Negotiation Protocol Discovery.

fully supports the role skeleton. Rule (3) states that there is a possible delegation D for a role skeleton S , where the delegator is actor A and the delegatee is actor A_2 , when actor A cooperates with actor A_2 and the latter actor supports the role skeleton. Rule (4) states that there is a possible cooperation C between a set of actors L for a role skeleton S , where the actor A is involved in it, when the set of actors L support the role skeleton S and all actors X of L that are different from A can cooperate with A .

Candidate protocols for negotiation are inferred through rule (5), which states that a protocol P is a candidate for negotiation N , when there is a list of role skeletons LS , filled in with a role skeleton S required by P , which is supported by a role R , and the list LS eventually contains all the possible role skeletons S required by protocol P . A role of a negotiation supports a role skeleton when all of the participants X that take part in the negotiation with this role can support, in any possible way, this role skeleton. In other words, the goal of rule (5) is threefold: a) to assign to the participants taking the same role in the negotiation the same role skeleton; b) to assign different role skeletons to participants of different roles; c) to assign all role skeletons of a negotiation protocol. Thus, when this goal is satisfied, the negotiation protocol can be used for enacting the negotiation.

There are three different cases in which an actor X can support a role skeleton S of protocol P . In the first case, the actor X directly supports (by himself or through his or her agent) the role skeleton S . In the second case, the actor X delegates all the actions of role skeleton S to a cooperating actor X_2 (the delegatee) through a possible delegation D . This can happen only if the actor X_2 does not participate in the same negotiation where X participates. In the third

case, the actor X participates in a possible cooperation C that supports role skeleton S . Also in this case, all the cooperating actors except X must not participate in the negotiation N where X participates. It should be noted that although rule (5) is very general, it can be used only in cases where the role skeletons in a negotiation protocol are mutually exclusive, that is, a role skeleton may not substitute another role skeleton in a negotiation and a participant in a negotiation, possibly through delegation/cooperation, can support only a single role skeleton in a given negotiation. Future work will concern the addition of further set of rules to support different types of negotiation protocols with different requirements for their roles skeletons.

Rules (6)–(9) support the inference of property *supportSkel* for rule (4). Rule (6) states that an actor partially supports a role skeleton when the actions he or she is able to perform do not include all the actions required by role skeleton. Rule (7) states that an actor list partially supports a role skeleton, with a set of actions L , when it contains only one actor (that is able to perform the L actions in the action list) and this actor partially supports the role skeleton. Rule (8) states that an actor list AL_2 partially supports a role skeleton S and supports a set of actions L_4 , when it is produced by an actor list AL_1 that partially supports S and an actor A that also partially supports S and the union L_4 of the action sets that AL_1 and A support does not include all the actions that S requires. Finally, rule (9) states that an actor list AL_2 supports a role skeleton S and supports a set of actions L_4 , when it is produced by an actor list AL_1 that partially supports S and an actor A that also partially supports S and the union L_4 of the action sets that AL_1 and A support is a super-set of the action set that S requires.

It should be noted that we have not illustrated some additional rules that either describe the creation and comparison of lists or the enforcement of some negotiation constraints due to the space limitation of this paper. An example of the latter type of rules is a rule forbidding an actor to have two or more representations in a specific negotiation.

Reasoning Complexity: Our developed ontology does not use the OWL-DL's features of nominals and cardinality constraints so it is a syntactic variant of the *SHI(D)* Description Logic (DL). However, we intend to enrich it with cardinality constraints, so it would eventually be a syntactic variant of the *SHIN(D)* DL. Both these DLs have an ExpTime-complete complexity for reasoning about concept satisfiability and ABox consistency.

Now, as we enrich our ontology with FOL rules, we will have an undecidable reasoning process. For this reason, we intend to translate the FOL rules into SWRL¹ ones. In this way, by combining OWL-DL with DL-Safe SWRL rules [12], we reach a decidable reasoning process.

¹www.w3.org/Submission/SWRL/

A. Examples of negotiation protocol discovery

In our example, we consider the case of bilateral negotiation between two actors, i.e. a service provider p and a service requester c . In order to show the reasoning capabilities of our framework, we also assume the existence of two assisting actors h_1 and h_2 , that can help the service provider and requester by extending their capabilities in order to support the two role skeletons of the bilateral negotiation protocol. For a better presentation, this example is separated into three parts. In the first part, we describe the bilateral negotiation protocol and its involved role skeletons. In the second part, we describe the capabilities/actions of each actor. Finally, in the third part we illustrate how our ontology model captures the appropriate knowledge for performing the negotiation protocol discovery .

Bilateral negotiation.: This negotiation protocol involves two actors, i.e. a service provider and requester. The following conditions apply for the enactment of the protocol:

- One of the actors is able to generate an offer from scratch, i.e. the first offer in a negotiation;
- Both actors should be able to send and receive offers;
- Both actors should be able to generate counteroffers, given an offer made by the counterpart;
- At least one of the actors is able to accept an offer and generate the outcome of a negotiation;
- Both actors are able to receive the outcome of the negotiation.

Hence, we can identify three role skeletons involved in this protocol:

- *Initiator*, able to generate the first offer.
- *Participant*, able to send and receive offers, to generate counter offers, and to receive the outcome of the negotiation.
- *Decision maker*, able to accept an offer and generate the outcome of a negotiation.

One and only one of the actors must be able to support the first role skeleton (Initiator). Both actors should support the second role skeleton (Participant), whereas at least one of the actors should support the third role skeleton (Decision Maker). Thus, there are some rules that govern the way the role skeletons of a negotiation protocol should be supported by the negotiating actors.

As already introduced while discussing rule (5), we consider exclusive role skeletons (i.e. one role skeleton must be supported by only one negotiating actor). For this reason, the new role skeletons of this negotiation protocol are:

- *Initiator*: Must be able to generate the first offer (Action co), to send and receive offers (Actions so and ro), to generate counter offers (Action cco) and to receive the outcome of the negotiation (Action go).
- *Participant*: Must be able to send and receive offers, to generate counter offers, and to generate the outcome of the negotiation (Action sto).

Initial Facts Entered (describing actions, protocols and capabilities of actors):
 Action(co), Action(so), Action(ro), Action(go), Action(ao), Action(sto),
 Actor(p), ableToPerform(p , [co , so , ro , go]), Actor(h_1), ableToPerform(h_1 , [cco , so , ro]),
 cooperatesWith(p , h_1), Actor(c), ableToPerform(c , [cco , ro , so , ao , sto]), Actor(h_2),
 ableToPerform(h_2 , [cco , ro , so , ao , sto]), cooperatesWith(c , h_2), BilateralNegotiation(bn),
 Initiator(i), performs(i , [co , cco , so , ro , go]), Participant(pa),
 performs(pa , [cco , ro , so , ao , sto]), Role(provider), Role(requester)

↓ (Rules 1 - 4 and 5 - 9 fired creating new facts)
 fSupportsSkel(c , pa), fSupportsSkel(h_2 , pa), PossibleDelegation(d), canDelegate(c , d),
 delegatee(d , h_2), skel(d , pa), pSupportsSkel(p , i), pSupportsSkel(h_1 , i),
 pSupportSkel(p , i), actionsSupported($[p]$, [co , so , ro , go]), supportSkel($[p]$, h_1 ; i),
 actionsSupported($[p]$, h_1), [co , cco , so , ro , go]), PossibleCooperation(pc),
 inCooperation(p , pc), cooperators(pc , [p , h_1]), ofSkeleton(pc , i)

↓ New Facts Entered (indicating the negotiation between p and c):
 Negotiation(n), participatesIn(p , n), takesRole(p , provider), participatesIn(c , n),
 takesRole(c , requester), ofNegotiation(provider, n), ofNegotiation(requester, n)

↓ (Rule 5 is fired and creates the desired fact)
 candidateProtocol(n , bn)

Figure 4. Facts and their inferencing.

Capabilities of Actors: We assume the following facts for our example:

- The actor p participates in the negotiation and takes the role of the service *provider*. The actor p is able to perform the actions co , so , ro , go .
- The actor c participates in the negotiation and takes the role of the service *requester*. The actor c is able to perform the following actions: cco , so , ro , ao , sto .
- The actor h_1 does not participate in the negotiation and is able to perform the actions cco , so , ro .
- The actor h_2 does not participate in the negotiation and is able to perform the actions cco , so , ro , ao , sto .

Reasoning: The service provider p does not fully support the *Initiator* role skeleton, but it may support it through the cooperation with actor h_1 . The service requester c fully supports the *Participant* role skeleton, but it may also support it through a full delegation to actor h_2 . Thus, the negotiation n between p and c actors can be enacted through the use of the bilateral negotiation protocol. This is ensured by rule (5). In the following, we illustrate the process where facts are entered and new facts are produced from the set of rules (1)-(9).

Initially, only the facts regarding the description of the actors and the negotiation protocol are entered in the ontology model. Then the majority of rules fire and new facts are produced, detecting if each actor can support the role skeletons of a protocol irrespective of the negotiations in which it is going to participate. Then new facts are entered in the model that describe the negotiation that is going to take place between the two actors. Finally, based on all previous facts, rule (5) is fired and the fact that the negotiation can use the bilateral negotiation protocol is produced. The whole process is depicted in Fig. 4 along with all the facts entered and derived.

V. RELATED WORK

The need for automated negotiation of quality SLAs and contracts, is being addressed as one of the main driver

for the adoption of service based systems in real-world scenarios [13]. However, the literature has usually focused only on the negotiation enactment phase. In this context, we identify two different approaches for the enactment of Web service QoS negotiation, i.e., the broker-based and the agent-based approaches. With the broker-based approach, the execution of negotiation is delegated by service providers and customers to a trusted broker [14]. Conversely, in the agent-based approach each negotiation actor defines its own agent, which embeds the actor's negotiation strategies [15], [16]. In both cases, the structure of the negotiation protocol is usually known a priori by the participants. Moreover, the ability of the broker to execute a protocol or of negotiation agents to enact a given negotiation protocol is taken for granted.

Activities and roles within a generic negotiation protocol are discussed in [9] in the context of negotiation support systems design, while a theoretical analysis of customizable generic negotiation protocols is presented in [17]. In particular, the latter work can be used in future extensions of our approach to give solid theoretical basis to the identification of relevant activities and roles within a given negotiation protocol.

Research on generic or semantic-based negotiation protocols belongs mostly to the area of agent-based computing. In this context, the work presented in [10] discusses an agent-based software framework for defining and executing negotiation protocols. The focus of the work is on the definition of a set of rules, from which several protocols, such as auctions or bilateral negotiations, can be instantiated. Also in this case, the framework does not focus on the description of agents negotiation capabilities and takes for granted the ability of negotiation agents to participate in an instantiated negotiation protocol. In [18] Chiu et al. discuss how an ontology can be helpful for supporting the negotiation. In particular, the authors highlight how shared and agreed ontologies provide common definitions of the terms to be used in the subsequent negotiation process. In addition, they propose a methodology to enhance the completeness of issues in requirements elicitation. Lamparter et al. [19] introduce a model for specifying policies for automatic negotiation of Web services. In this case, the starting point is the upper ontology DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) [20]. On this basis, this work proposed a policy ontology that also includes the user preferences specifications and an additional ontology for expressing the contracts.

About the use of ontology for specifying the agreement among parties, Oldham et al. [21] present reasoning methods for the components of an agreement which must be compatible for quality matches. This approach is based on WS-Agreement and takes advantage of Semantic Web methods to achieve rich and accurate matches. With the same goal Uszok et al. [22] have developed KAoS policy ontology that

allows for the specification, management, conflict resolution, and enforcement of policies within the specific contexts established by complex organizational structures.

VI. CONCLUDING REMARKS

In this paper we introduced a framework based on an ontology to support the negotiation in a SOA. Using this framework service requesters and providers are able to describe which are the negotiation protocol they are able to carry out and what are the actions they can perform. In case a negotiation is required, by an ontology reasoning, the framework can identify which are the actors involved and according to which protocol a negotiation can be executed.

At this stage the framework is mainly focused on the identification of the set of negotiation protocols that can be potentially executed. Future work will be focused on an extension able to select the best negotiation protocols among the discovered ones. In addition, a broader analysis of the impact of this framework in the Service Oriented Architecture has to be analyzed in terms of cost/benefit analysis. Indeed, on the one hand, our framework has the main advantage of making possible the communication of services that could not previously occurred due to the unfeasibility of the negotiation. On the other hand, the effort of describing capabilities and requirements along with the time for executing the discovery process needs to be considered, too.

ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube) and the Italian FIRB Project TEKNE

REFERENCES

- [1] J. Garofalakis, Y. Panagis, E. Sakkopoulos, and A. Tsakalidis, "Contemporary Web service discovery mechanisms," *Journal of Web Engineering*, vol. 5, no. 3, pp. 265–290, 2006.
- [2] K. Kritikos and D. Plexousakis, "Semantic qos metric matching," in *ECOWS '06: Proceedings of the 4th European Conference on Web Services*, 2006, pp. 265–274.
- [3] M. P. Papazoglou and G. Georgakopoulos, "Service-oriented computing: Introduction," *Communication ACM*, vol. 46, no. 10, pp. 24–28.
- [4] A. R. Lomuscio, M. Wooldridge, and N. R. Jennings, "A classification scheme for negotiation in electronic commerce," *Group Decision and Negotiation*, vol. 12, pp. 31–56, December 2003.
- [5] P. Plebani and B. Pernici, "Urbe: Web service retrieval based on similarity evaluation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 99, no. 1, 5555.
- [6] K. Kritikos and D. Plexousakis, "Requirements for qos-based web service description and discovery," in *Proc. 31st Annual Int. Computer Software and Applications Conference (COMPSAC 2007)*, 2007, pp. 467–472.
- [7] M. Comuzzi, K. Kritikos, and P. Plebani, "Semantic-aware service quality negotiation," in *Proc. ServiceWave. LNCS 5377*, Madrid, Spain, December 2008, pp. 312–323.
- [8] A. Vedamuthu, D. Orchard, F. Hirsch, M. Hondo, P. Yendluri, T. Boubez, and U. Yalcinalp, "Web Services Policy 1.5 - Framework (W3C Recommendation)," <http://www.w3.org/TR/ws-policy>, September 2007.
- [9] M. Bichler, G. Kersten, and S. Strecker, "Towards a structured design of electronic negotiations," *Group Decision and Negotiation*, vol. 12, pp. 311–335(25), July 2003.
- [10] C. Bartolini, C. Preist, and N. R. Jennings, "Architecting for reuse: A software framework for automated negotiation," in *in Proc. 3rd Int Workshop on Agent-Oriented Software Engineering*, 2002, pp. 87–98.
- [11] D. Trastour, C. Bartolini, and C. Preist, "Semantic web support for the business-to-business e-commerce pre-contractual lifecycle," *Computer Networks*, vol. 42, no. 5, pp. 661–673, 2003.
- [12] B. Motik, U. Sattler, and R. Studer, "Query answering for owl-dl with rules," in *ICWS 2004: Third International Semantic Web Conference*, vol. 3298. Hiroshima, Japan: Springer, 2004, pp. 549–563.
- [13] M. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-Oriented Computing: State of the art and research challenges," *IEEE Computer*, vol. 11, pp. 38–45, 2007.
- [14] M. Comuzzi and B. Pernici, "An architecture for flexible Web service QoS negotiation," in *Proc. 9th IEEE Enterprise Computing Conference*, Enschede, The Netherlands, 2005.
- [15] M. B. Chhetri, J. Lin, S. Goh, J. Y. Zhang, R. Kowalczyk, and J. Yan, "A coordinated architecture for the agent-based service level agreement negotiation of web service composition," in *Proc. Australian Software Engineering Conference*, 2006, pp. 90–99.
- [16] E. Di Nitto, M. Di Penta, A. Gambi, G. Ripa, and M. L. Villani, "Negotiation of service level agreements: An architecture and a search-based approach," in *In Proc. ICSOC'07*, 2007, pp. 295–306.
- [17] G. Kersten, S. Strecker, and K. Law, "Protocols for electronic negotiation systems: Theoretical foundations and design issues," in *Proc. EC-Web 2004 (LNCS 3182)*, 2004, pp. 106–115.
- [18] D. K. W. Chiu, S. C. Cheung, P. C. K. Hung, and H. fung Leung, "Facilitating e-negotiation processes with semantic web technologies," in *Proc. 38th Annual Hawaii International Conference on System Sciences*, 2005.
- [19] S. Lamparter, S. Luckner, and S. Mutschler, "Formal specification of web service contracts for automated contracting and monitoring," in *Proc. 40th Annual Hawaii International Conference on System Sciences*, 2007.
- [20] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, A. Oltramari, and L. Schneider, "Wonderweb deliverable d17. the wonderweb library of foundational ontologies and the dolce ontology." [Online]. Available: citeseer.ist.psu.edu/masolo02wonderweb.html
- [21] N. Oldham, K. Verma, A. Sheth, and F. Hakimpour, "Semantic ws-agreement partner selection," in *WWW 2006*, 2006, pp. 697–706.
- [22] A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni, and J. Lott, "Kaos policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement," in *Proc. 4th IEEE Int. Workshop on Policies for Distributed Systems and Networks*, 2003.