A semantic based framework for supporting negotiation in Service Oriented Architectures

Kyriakos Kritikos and Pierluigi Plebani
Politecnico di Milano
&
Marco Comuzzi
City University London
Agenda

- Reference scenario
- The problem
- The negotiation framework
- An example in case of bilateral negotiation
- Concluding remarks and future work
Introduction

- **Facts:**
  - Service discovery produces a set of available services for each requester
  - Negotiation is required before (or after) selecting the best service for each requester and establish the SLA

- **Problems:**
  - Current solutions assume all negotiation participants agree and are able to support the same negotiation protocol
    - Under a closed-world assumption the negotiation protocol can be selected at design time
    - Under a open-world assumption the negotiation protocol should be selected at run-time
Focus of this work

[Diagram of service broker, service discovery, negotiation protocol discovery, service selection, service execution processes involving service requesters and service providers with annotations like Actor, Activity, step, data, find, publish, invoke, execute, and negotiation protocol selection.]
Starting point

In our previous work [1], we extended OWL-Q [2] by focusing:

- on the *negotiation objects*
- on the negotiators’ *decision models*

in order to:

- automate the service quality specification alignment and matchmaking processes
- assist the actual negotiation process through reasoning with rules
  - showed how to infer *negotiation compatibility* of service quality offers and demands
On OWL-Q and its negotiation extension

- OWL-Q is an upper ontology complementing OWL-S
- It comprises of many sub-ontologies/facets
- Each facet concentrates on a particular aspect of the QoS modeling and can be extended independently of the others
Main idea

- Participants should advertise their negotiation capabilities in a more fine-grained way

- Matching of negotiation capabilities and of actions required by negotiation protocols is needed to find a negotiation protocol

- Three cases:
  - **Exact match**: a negotiation protocol is supported by all the participants
  - **Negotiation by delegation**: a negotiation protocol can be indirectly found if some negotiation participants can fully or partially delegate the missing capabilities to other cooperating actors
  - **Failed match**: a negotiation protocol cannot be found
Negotiation Conceptual Model (1/2)

- Negotiation actors: user or agents

- Role skeleton in bilateral negotiation:
  - **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.

- Actions in bilateral negotiation:
  - create offer (co)
  - send offer (so)
  - receive offer (ro)
  - create counter offer (cco)
Two types of delegation
- Full: An actor delegates all actions to another actor
- Partial: An actor delegates some of the actions to one or more cooperating actors

Two possible situations
- Federation: relationships among the actors are more strict; we can assume that actors trust in some way
- Marketplace: relationships among the actors are more loose; not all the data about negotiation could be exchanged
Implementation

- Semantic framework for negotiation protocol discovery, consisting of:
  - A reference ontology: a new OWL-Q extensions for describing the negotiation capabilities of the participants
  - A set of basic rules: for matchmaking the capabilities of the participants with the actions required by a negotiation protocol irrespective of any negotiation
  - A set of additional rules: for selecting a specific protocol when participants are involved in a negotiation
    - Only for bilateral negotiation until now but plan for all other common negotiation protocols
### Basic rules

1. $fSUPPORTSKEL(A, S) \leftarrow ableToPerform(A, L_1) \land performs(S, L_2) \land subList(L_2, L_1)$
2. $delegatesSkel(A, S) \leftarrow actsFor(A_2, A) \land fSUPPORTSKEL(A_2, S)$
3. $POSSIBLEDELEGATION(D) \land canDelegate(A, D) \land delegatee(D, A_2) \land skel(D, S) \leftarrow$
   $\leftarrow cooperatesWith(A, A_2) \land supportsSkel(A, S)$
4. $POSSIBLECOORDINATION(C) \land inCOORDINATION(A, C) \land cooperators(C, L) \land ofSkeleton(C, S) \leftarrow$
   $\leftarrow supportsSkel(L, S) \land contains(L, A) \land \forall X$ (member($L, X$) \land differentFrom($X, A$) $\rightarrow$
   $\rightarrow$ canCooperate($X, D$) \land delegatees($X, D$) \land skel($D, S$) \land performs($X, A$) \land
   $\land performs(D, A)$ \land ofSkeleton($C, S$) \land \exists X_2$ (member($L, X$) \land
differentFrom($X_2, X$) \land participatesIn($X_2, N$)) \land listAdd($L, S$))
   $\land \forall S$ (skeleton($P, S$) \land member($L, S$))
5. $CANDIDATEPROTOCOL(N, P) \leftarrow newList(LS) \land \forall R$ (ofNegotiation($R, N$) \land \exists S$ (skeleton($P, S$) \land
   $\land$ member($LS, S$) \land \forall X$ (participatesIn($X, N$) \land takesRole($X, R$) $\land$ supports($X, S$) $\vee$
   $\land$ canDelegate($X, D_1$) \land delegatees($D_1, X_2$) \land skel($D_1, S$) \land
   $\land$ participatesIn($X_2, N_2$)) $\land$
   $\land$ differentFrom($X_2, X$) \land participatesIn($X_2, N$))) \land listAdd($LS, S$))
   $\land \forall S$ (skeleton($P, S$) \land member($LS, S$))

### Rules for inferring the candidate protocols

6. $pSUPPORTSKEL(A, S) \leftarrow ableToPerform(A, L_1) \land performs(S, L_2) \land$
   listIntersection($L_3, L_1, L_2$) \land strictlySubList($L_3, L_2$)
7. $pSUPPORTSKEL(AL, S) \land actionsSupported(AL, L) \leftarrow pSUPPORTSKEL(A, S) \land$
   $\land$ ableToPerform($A, L$) \land listAdd($AL, A$)
8. $pSUPPORTSKEL(AL_2, S) \land actionsSupported(AL_2, L_4) \leftarrow pSUPPORTSKEL(AL_1, S) \land$
   $\land$ pSUPPORTSKEL($A, S$) \land ableToPerform($A, L_1$) \land
   $\land$ strictSubList($L_2, L_1$) \land Performs($S, L_2$) \land
   $\land$ listConcatenation($L_3, L_1, L_3$) \land
   $\land$ listIntersection($L_5, L_4, L_2$) \land
   $\land$ strictlySubList($L_5, L_2$) \land newList($AL_2, AL_1, A$)
9. $SUPPORTSKEL(AL_2, S) \land actionsSupported(AL_2, L_4) \leftarrow SUPPORTSKEL(AL_1, S) \land$
   $\land$ ($A, S$) \land ableToPerform($A, L_1$) \land
   $\land$ actionsSupported($AL_1, L_3$) \land
   $\land$ listConcatenation($L_4, L_1, L_3$) \land
   $\land$ performs($S, L_2$) \land subList($L_2, L_4$) \land
   $\land$ newList($AL_2, AL_1, A$)
Basic rules

\[ f\text{SupportsSkel}(A, S) \leftarrow \text{ableToPerform}(A, L_1) \land \text{performs}(S, L_2) \land \text{subList}(L_2, L_1) \]

An actor A fully support a skeleton S when he is able to perform all the actions \((L_1)\) that the role skeleton should perform \((L_2)\)

\[ \text{delegatesSkel}(A, S) \leftarrow \text{actsFor}(A_2, A) \land f\text{SupportsSkel}(A_2, S) \]

An actor A delegates a skeleton S when he has an agent \(A_2\) that sully supports the skeleton
Basic rules

candidateProtocol(N, P) ← newList(LS) ∧ ∀R(of Negotiation(R, N)) ∧ ∃S(skeleton(P, S)) ∧ ¬member(LS, S) ∧ ∀X(participatesIn(X, N)) ∧ takesRole(X, R) ∧ (supports(X, S) ∨ (canDelegate(X, D) ∧ delegatee(D, X_2) ∧ skel(D, S) ∧ ¬participatesIn(X_2, N)) ∨ (inCooperation(X, C) ∧ cooperators(C, L) ∧ ofSkeleton(C, S) ∧ ¬∃X_2(member(L, X) ∧ differentFrom(X_2, X) ∧ participatesIn(X_2, N)))) ∧ listAdd(LS, S)) ∧ ∀S(skeleton(P, S) ∧ member(LS, S))

• to assign to the participants taking the same role in the negotiation the same role skeleton;
• to assign different role skeletons to participants of different roles;
• 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.
Basic rules

\[ \text{supportSkel}(AL_2, S) \land \text{actionsSupported}(AL_2, L_4) \leftarrow \text{pSupportSkel}(AL_1, S) \land \]
\[ \land (A, S) \land \text{ableToPerform}(A, L_1) \land \text{actionsSupported}(AL_1, L_3) \land \]
\[ \land \text{listConcatenation}(L_4, L_1, L_3) \land \text{performs}(S, L_2) \land \text{subList}(L_2, L_4) \land \text{newList}(AL_2, AL_1, A) \]

- 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.
Example for bilateral negotiation

- 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 $cco, so, ro, ao, sto$. 
Example for bilateral negotiation

- The service provider p does not fully support the Initiator role skeleton, but it may support it through the cooperation with actor h1.

- The service requester c fully supports the Participant role skeleton, but it may also support it through a full delegation to actor h2.
Example for bilateral negotiation

Negotiation(n), participatesIn(p, n), takesRole(p, provider), participatesIn(c, n), takesRole(c, requester), of Negotiation(provider, n), of Negotiation(requester, n)

candidateProtocol(n, bn)

- Thus, the negotiation n between p and c actors can be enacted through the use of the bilateral negotiation protocol.
Future Work

- Develop a set of rules for matchmaking the most common negotiation protocols apart from bilateral negotiation
- Extend our framework for selecting the best negotiation protocol among the candidate ones
- Analyze how to involve humans in the negotiation
- Create and evaluate a prototype implementation of our framework
References
