An Adaptive Ontology-Mediated Approach to Organize Agent-based Supply Chain Negotiation

Gong Wang, T N Wong, Xiaohuan Wang
Department of Industrial and Manufacturing Systems Engineering
The University of Hong Kong
Hong Kong
wanggong@hkusua.hku.hk

Abstract— **Effective supply chain management (SCM) comprises activities involving the demand and supply of resources and services. One important aspect of SCM is that companies in the supply chain may have to make decisions which are conflicting with the other partners. Negotiation is an essential approach to solve transaction and scheduling problems among supply chain members. Multi-agent systems (MAS) are being increasingly used in SCM applications. The advances in agent technology have provided the potential of automating supply chain negotiations to alleviate human interactions. This paper proposes an ontology-mediated approach to organize the agent-based supply chain negotiation and equip the agents with sophisticated negotiation knowledge. Firstly, a generic agent negotiation scheme is developed involving the agent intelligence modules, the knowledge representation method and the interaction behaviors. Then, the negotiation knowledge is structured through the usage of ontology, which performs as a hierarchical architecture as well as a descriptive language. The relationships between negotiation ontology concepts are defined through SWRL inference rules. Through this method, agents' negotiation behaviors will be more adaptive to various negotiation environments in accordance with different negotiation knowledge.**

Keywords-supply chain; negotiation; multi-agent; ontology; inference rules

I. INTRODUCTION

A supply chain is a network of interconnected supply chain members participating in the production, delivery and retail of a particular product. Effective supply chain management (SCM) involves activities to solve the conflicts between demand and supply of resources and services. For supply chain members with conflicting interests or viewpoints, negotiation is an essential approach for decision making and reaching the mutual agreement. Negotiations in supply chains may relate to a wide range of details in transaction processes, including the product specification, cost and pricing policy, trade terms and so on. During the negotiation process, the reaction of the negotiation opponents and the dynamics of the market circumstances should be captured.

As a novel approach of business automation, multi-agent systems (MAS) are being increasingly used in SCM applications, involving generic supply chain modeling [1] and frameworks for specific industrial areas [2]. With the frequently mentioned features of autonomous, cooperative and reactive, the MAS can provide the potential of automating supply chain negotiations. Ideally, this approach will shorten the time consumed in tedious negotiation processes and alleviate the human interactions. For current researches on agent-based negotiation models, Jonker et al. [3] developed a one-to-one multi-attribute negotiation model in which agents are able to use incomplete preference information revealed by the negotiation partner. To some extent, this generic agent architecture reflects the supply chain negotiation realities, in which supply chain members are prone to keep their privacy as much as desired. Wang et al. [4] proposed an argumentation-based multi-agent negotiation model for dynamic supply chain formation. In this model, the major functions of the negotiation agents are generating and exchanging information to update agents' understanding of the preferences and conflicts. As a general concern, in agent-based supply chain negotiations, software agents have to be equipped with negotiation knowledge and preferences of supply chain members, and carry out negotiations through computing, reasoning and learning to make the best use of the available information.

To represent the knowledge and rules in agent-based systems, the concept of ontology [5] has been adopted in recent researches. By describing a set of concepts and the relationships between them, the ontology can construct both the hierarchical architecture of the negotiation knowledge and the descriptive logics of negotiation rules and activities. Some ontology-mediated approaches have been proposed to make agents adapt to various negotiation mechanisms by representing the negotiation protocol in an ontological manner [6, 7]. There are also works focusing on the heterogeneity problems between agents using different ontologies in the dynamic open environment [8]. In these researches, ontologies are represented via Web Ontology Language (OWL), which is a markup langrage with richer expressive power. Ideally, these approaches can ensure the communications and interoperability between agents. Then, the other concern is to guide the agents' negotiation behavior in accordance with concepts and relationships defined in the ontology architecture. That is to say, agents in the interaction procedure should decide or reason about their negotiation behavior automatically when the current negotiation circumstance is reflected to the ontology architecture. It is therefore necessary to define logic rules on top of the ontology architecture to extend and activate the reasoning ability of agents. To achieve compatibility in the development of rules and ontologies, the Semantic Web Rule

Language (SWRL) can be used to develop the logic rules and combine with the OWL ontology expressions.

In this paper, an ontology-mediated approach is proposed to organize the agent-based supply chain negotiation and to equip the agents with sophisticated negotiation knowledge. Logic rules are defined upon the negotiation ontology architecture to express the inference properties. Agents utilize the inference engine to adapt their negotiation behaviors to the specific negotiation situation. The rest of this paper is structured as follows. In section 2, the generic multi-agent negotiation scheme in supply chains is described. Section 3 illustrates the design of negotiation ontology architecture and ontology inference rules. In section 4, the proposed negotiation ontology and its instantiations are created using software tools. The rules engine is tested to infer negotiation behaviors. Finally, conclusions are drawn in section 5.

II. Multi-Agent Negotiation Scheme in Supply Chains

In supply chains, the transaction of resources, products and services forms the major linkage between supply chain members. Similar to the bargaining of price between the buyer and seller in a marketplace, negotiation in a supply chain is an essential approach for the transaction counterparts to solve their conflicting interests. What makes the negotiation scenario more complicated in supply chains is that, multiple supply chain members may involve in the transaction process with multiple negotiation issues to be considered. That is to say, negotiations in supply chains are usually multi-lateral and multi-issue negotiations.

In this paper, supply chain negotiation activities are abstracted into two agent roles, namely, the buyer agent and seller agent. During the negotiation process, supply chain partners are represented by either the buyer or seller agent according to their respective roles and functions. The complex multi-lateral negotiation is to be modeled by a number of bilateral buyer-seller negotiations. The basic negotiation functions are structured and defined within the bilateral buyer-seller negotiation scheme. Figure 1 depicts the buyer-seller agent negotiation scheme showing the intelligence modules inside agents, the knowledge representation method and the interaction behaviors in negotiations.

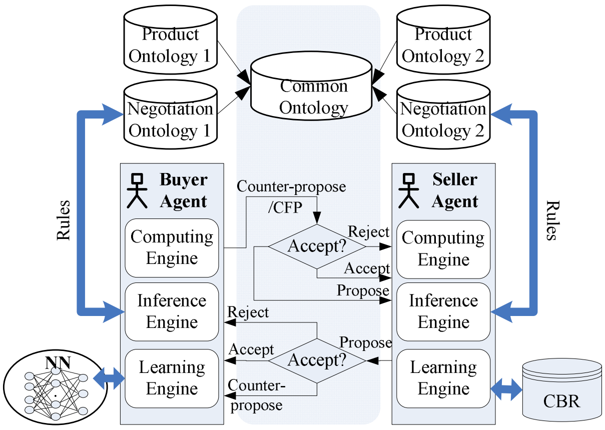


Figure 1. Buyer-seller agent negotiation scheme.

Three intelligence modules are built in both buyer and seller agents to facilitate their dynamic negotiation behaviors and decision makings. The computing engine is encoded in software agents to generate the negotiation issue values and calculate the acceptance level of the opponent's proposal. The inference engine is based on the logic rules defined in negotiation ontologies. Through the inference engine, agents can dynamically adapt their negotiation strategies to specific negotiation situations. The learning engine is embedded to enable agents' abilities of learning and predicting opponent's negotiation behaviors. The learning engine may be based on the Neural Network (NN), Case-Based Reasoning (CBR) or other potential learning methods. Obviously, an agent will be in a better position if it is able to learn more from the opponent's behavior during the negotiation interactions.

The negotiation knowledge, involving the specification of negotiation issues, the information of negotiation partners, the negotiation strategies to make concessions and so on, is represented as concepts in the negotiation ontology. The direct and indirect relations between the concepts are constructed as logic rules on the top of the ontology architecture. Details of the ontology design will be illustrated in the next section. In order to ensure the mutual understanding between agents using different ontologies, the agent's individual negotiation ontology and other related ontologies (such as product ontology) have to be combined or mapped into a common ontology. With the establishment of the common ontology, ideally, there will be no misunderstandings between the buyer and seller agents.

For the interaction behaviors in the negotiation procedure, messages are exchanged between agents to deliver their proposals on the negotiation issues. Terms used in the proposals are based on the common ontology. The negotiation may start from the buyer agent's call for proposal (CFP) message to announce its demand to the sellers. During the negotiation process, the seller and buyer agents will send their proposals and counter-proposals alternately and iteratively until the mutual agreement is achieved. For each proposal or counter-proposal, the buyer or seller agent has to decide its action to reject, accept or make concession on the negotiation issues.

With regard to the generic buyer-seller negotiation scheme, the multi-lateral negotiation scenarios along the supply chain can be organized as illustrated in figure 2. The coordinator agent (CA) is introduced here to control multiple buyer agents (BA) and seller agents (SA). As an example, if Manufacturer1 needs to acquire a certain resource, it has to negotiate with potential suppliers to select the best one. Manufacture1 will be represented by BAs. The CA in Manufacturer1's system will govern individual BAs to carry out buyer-seller negotiations with different suppliers' SAs and finally select the best deal. On the other hand, if Manufacturer1 needs to sell products to retailers, it is then represented by SAs. The CA will govern individual SAs to carry out bilateral negotiations with retailers' BAs. Similar configurations can be applied upstream or downstream the supply chain when transactions take place.

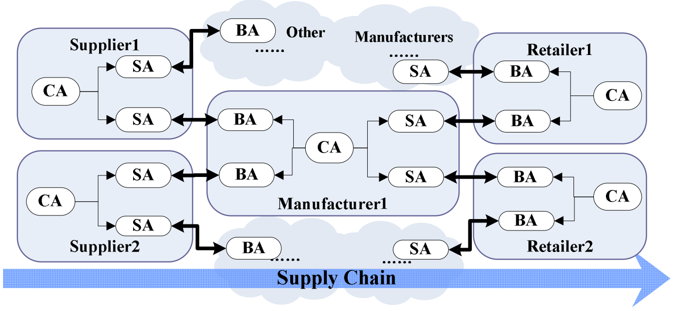


Figure 2. The organization of multi-lateral negotiation.

III. The Negotiation Ontology System Design

The negotiation ontology can be perceived as task ontology according to some ontology classification method. It involves the knowledge required to fulfill the negotiation tasks. There are two steps to construct expressive negotiation ontology. The first is to design the ontology architecture dealing with the elicitation of basic concepts in the negotiation knowledge, and the specification of properties and constrains in these concepts. The second is to define the logical axioms or rules which can describe the relationships between concepts and properties.

A. Ontology Architecture Design

To conduct the agent-based negotiation, three aspects of knowledge are required to be considered, namely, negotiation issues, negotiation protocol and negotiation strategies [9]. The negotiation issues clarify the scope of communicating. The negotiation protocol regulates the encounter rules. The negotiation strategies outline the concession behaviors agents can perform. Hence, the primary negotiation ontology concepts are constructed to reflect theses considerations. Figure 3 shows the architecture of the negotiation ontology, in which six concepts are described using specific properties. Since the negotiation issue specifications may relate to terminologies from other domain ontologies, the Order and Product concepts are attached to the negotiation ontology with dashed lines as an indication. The definitions of the concepts are briefly introduced below.

The Issue concept is defined to configure any concerning object in transactions. The forNegotiation property denotes whether the issue is negotiable. The hasType property indicates the measuring type (cost or benefit) of the issue if applicable. The hasWeighting property reflects the relative importance of the issue among the overall concerning objects. Other properties are used to describe the value range of the issue.

The AgentIdentifier concept outlines the general profiles of agents participating in the negotiation, including their roles and historical performance information.

The Protocol concept regulates the interaction patterns according to the negotiation environment settings (such as the number of buyers and sellers, and negotiation issues).

The NegotiationStatus concept summarizes the current negotiation situation, involving the negotiation participants, negotiation issues, negotiation deadline and so on.

The Strategy concept is defined to facilitate the adaptive negotiation behavior configuration. Since the calculation functions of negotiation strategies are encoded within agents, this concept will provide the specific function parameters (concession trend, concession convexity and concession speed) based on the current negotiation situation.

The IssueStatus concept reflects the dynamics of the negotiation opponent's behavior in line with the agent's detecting results during the negotiation procedure.

This ontology architecture has been coded using OWL so as to be accessible by software agents. In table 1, the OWL code segments for the Issue concept are provided to express the property in the concept, the constraint of the property and the instantiation individual of the concept.

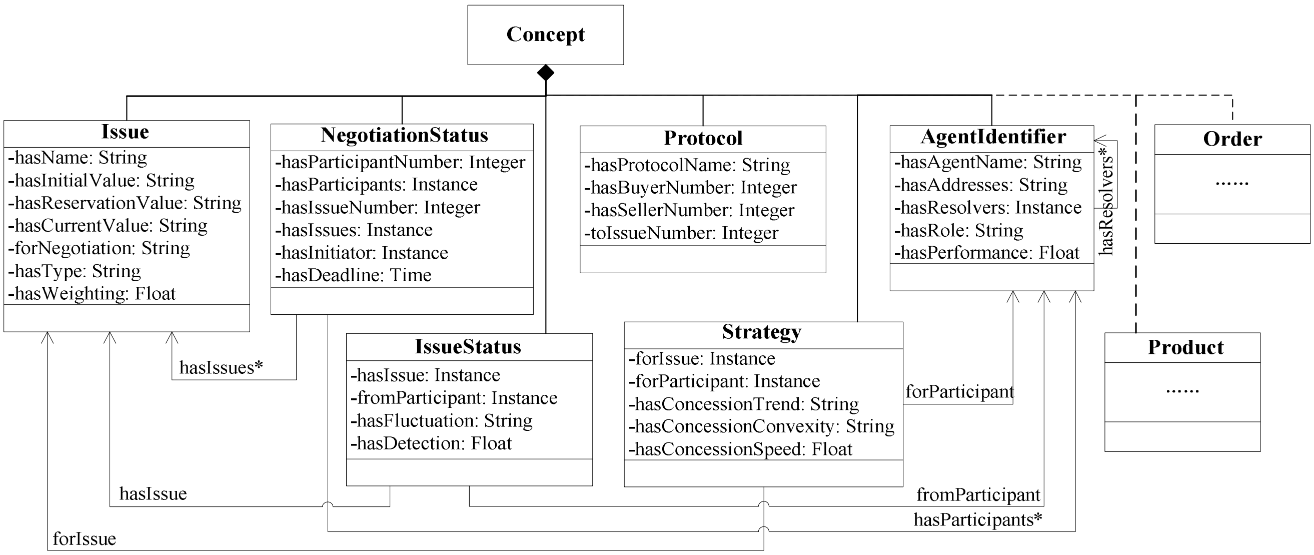


Figure 3. The negotiation ontology architecture.

TABLE I. OWL CODE SEGMENTS FOR THE ISSUE CONCEPT

Property

```
<owl:DatatypeProperty rdf:ID="hasType">
    <rdfs:domain rdf:resource="#Issue"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
```

Constraint

```
<owl:Class rdf:ID="Issue">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasType"/>
      <owl:cardinality rdf:datatype="&xsd;int">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
    ……
</owl:Class>
```

Individual

```
<Issue rdf:ID="Price">
    <hasType rdf:datatype="&xsd;string">cost</hasType>
    <forNegotiation
        rdf:datatype="&xsd;string">yes</forNegotiation>
    <hasReservationValue
        rdf:datatype="&xsd;string">80</hasReservationValue>
    <hasInitialValue
        rdf:datatype="&xsd;string">50</hasInitialValue>
    <hasName rdf:datatype="&xsd;string">price</hasName>
    <hasCurrentValue
        rdf:datatype="&xsd;string">50</hasCurrentValue>
    <hasWeighting rdf:datatype="&xsd;float">0.6</hasWeighting>
</Issue>
```

B. *Ontology Rules Definition*

The ontology architecture can be expressed using OWL, but the relationships between concepts, properties and individuals are yet to be clarified. The SWRL rules provide procedural knowledge, which compensates for some of the limitations of ontology inference, particularly in identifying semantic relationships between individuals [10]. SWRL utilizes a human readable logic expression "antecedent→consequent" to express rules or axioms. Here, both the antecedent and consequent are conjunctions of atoms written as $a_1 \wedge \ldots \wedge a_n$. Variables are indicated using the convention of prefixing them with a question mark (e. g., ?x).

In the negotiation ontology, there are direct or indirect relationships between the properties of concepts. It means that the properties of one concept may influence the configuration of other concepts. Only in this manner, agents can infer their negotiation behaviors based on the overview of the negotiation environment. For example, as a general knowledge, if an issue is negotiable, then the negotiation will include this issue. Representing this consideration through SWRL, a logic rule can be defined as (1):

NegotiationStatus(?x) ∧ Issue(?y) ∧ forNegotiation(?y, ?z) ∧ swrlb:equal(?z, "yes")→hasIssues(?x, ?y) (1)

Generally, the SWRL inference rules can be developed from the human comprehensible "if…then…" logic which indicates the scenario of if some conditions are satisfied, then some results can be inferred. Table 2 shows two examples of conversions between if-then logics and SWRL rules. These examples are assumed to infer the strategy properties from other properties in the negotiation ontology.

TABLE II. SWRL RULES DEFINITION

If-then logic	SWRL rules
If the negotiation initiator is a buyer and the negotiation issue is a cost type one, then the initiator's concession trend for this issue is increased.	Strategy(?x)∧forIssue(?x,?d)∧ hasName(?d,?e)∧ NegotiationStatus(?y)∧hasInitiator(?y,?z) ∧hasRole(?z,?a)∧ swrlb:equal(?a,"buyer")∧hasIssues(?y,?b) ∧hasType(?b,?c)∧ hasName(?b,?f)∧swrlb:equal(?c, "cost")∧ swrlb:equal(?e,?f)→ hasConcessionTrend(?x,"increased")
If the concession speed is greater than 1, then the concession function is concave.	Strategy(?x)∧hasConcessionSpeed (?x,?y)∧swrlb:greaterThan(?y,1)→ hasConcessionConvexity(?x, "concave")

Since the SWRL rules are defined on the top of the ontology architecture, they can reason about OWL individuals primarily in terms of OWL classes and properties. Table 3 provides the OWL code segment of the SWRL rule (1).

However, the SWRL rules cannot be directly executed in a rule engine. The SWRL syntax must be converted into a comprehensible syntax in a rule engine, such as Jess (Java Expert System Shell) [6]. Jess rules have another syntax pattern in which the SWRL rule (1) takes the following format:

(defrule Rule-1 (forNegotiation ?y ?z) (NegotiationStatus (name ?x)) (Issue (name ?y)) (bindingFact ?f:0&:(invokeSWRLBuiltIn Rule-1 swrlb:equal ?z "yes")) => (assert (hasIssues ?x ?y)) (assertOWLProperty "hasIssues" ?x ?y)) (2)

TABLE III. OWL CODE SEGMENT FOR A SWRL RULE

SWRL rule

```
<swrl:Imp rdf:ID="Rule-1">
   <swrl:head>
         ……
   </swrl:head>
   <swrl:body>
       <rdf:List>
         <rdf:first>
           <rdf:Description>
             <rdf:type rdf:resource="&swrl;ClassAtom"/>
             <swrl:argument1 rdf:resource="#x"/>
             <swrl:classPredicate
                   rdf:resource="#NegotiationStatus"/>
           </rdf:Description>
         </rdf:first>
        <rdf:rest>
         <rdf:List>
          <rdf:first>
            <rdf:Description>
             <rdf:type rdf:resource="&swrl;ClassAtom"/>
             <swrl:argument1 rdf:resource="#y"/>
             <swrl:classPredicate rdf:resource="#Issue"/>
           </rdf:Description>
           ……
   </swrl:body>
</swrl:Imp>
```

The SWRL rules can be converted into Jess rules using an ontology editor (such as Protégé) and the SWRL and Jess plug-ins. Then, the Jess rules engine can be embedded in Java applications, such as JADE (Java Agent DEvelopment framework), and provide a flexible two-way run-time communication between Jess rules and agents.

IV. Primary System Implementation

For the multi-agent negotiation scheme in supply chains, the agent system is being developed in the JADE platform. The negotiation ontology architecture and SWRL rules are edited in Protégé (which is a prevalent ontology editor and knowledge-base framework developed by Stanford) combining with the SWRLTab (which is a development environment for SWRL rules). The SWRL rules are executed using the Jess rule engine through the SWRLJessTab plug-in, which can convert the SWRL rules to Jess rules.

A. Ontology Construction and Rules Engine Test

Before implementing the agent negotiation system, the negotiation ontology is built to organize the related negotiation knowledge, and the rules engine is run outside agents to test the inference potential.

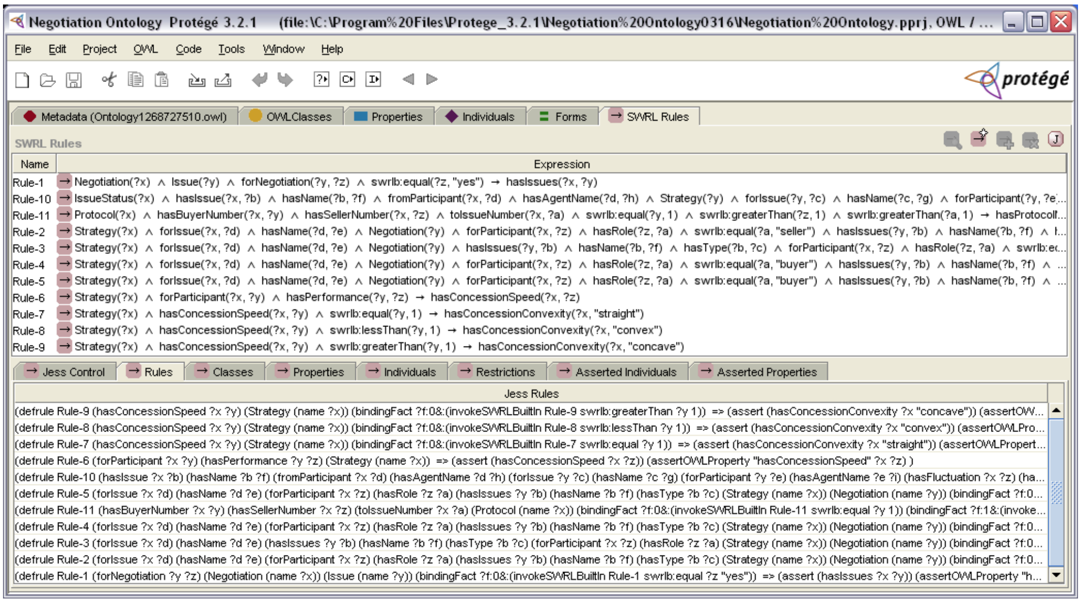
According to the ontology architecture illustrated in section 3, 6 classes, 12 individuals and 42 properties are constructed correspondingly. Initially, 11 SWRL rules are defined to represent some relationships between the classes and properties. Writing the OWL and SWRL to Jess rules, the Jess rules engine can be executed. Subsequently, 22 properties are inferred from the existing individual properties.

Figure 4 displays the SWRLTab window in Protégé. The upper part shows the 11 SWRL rules defined in the negotiation ontology, the lower part shows the result of converting the SWRL rules to Jess rules.

Some of the inference results are provided in table 4. The existing individual properties mainly denote the status of issues and negotiation agents. The inferred properties are generated by rules engine to deliver the strategy and protocol used in the specific negotiation situation.

TABLE IV. A Part of the Inference Result

Existing properties
(assert (hasName Price "price")) (assert (forNegotiation Price "yes")) (assert (hasType Price "cost")) (assert (hasName Quality "quality")) (assert (forNegotiation Quality "yes")) (assert (hasType Quality "benefit")) (assert (hasRole B "buyer")) (assert(hasAgentName B"B@828-research01:JADE/1099")) (assert (hasRole S1 "seller")) (assert(hasAgentName S1"S1@828-research01:JADE/1099")) (assert (hasRole S2 "seller")) (assert(hasAgentName S2"S2@828-research01:JADE/1099")) (assert (hasPerformance S1 4)) (assert (forIssue Strategy_1_1 Price)) (assert (forParticipant Strategy_1_1 S1)) (assert (toIssueNumber Protocol_1 2)) (assert (hasSellerNumber Protocol_1 2)) (assert (hasBuyerNumber Protocol_1 1)) (assert (fromParticipant IssueStatus_1 S1)) (assert (hasFluctuation IssueStatus_1 "drastic")) (assert (hasDetection IssueStatus_1 2))
Inferred properties
(assert (hasIssues Negotiation_1 Price)) (assert (hasIssues Negotiation_1 Quality)) (assert (hasProtocolName Protocol_1 "multi-issue reverse auction")) (assert (hasConcessionTrend Strategy_1_1 "increased")) (assert (hasConcessionSpeed Strategy_1_1 "4")) (assert (hasConcessionConvexity Strategy_1_1 "concave")) (assert (hasConcessionSpeed Strategy_1_1 "2"))



Name	Expression
Rule-1	Negotiation(?x) ∧ Issue(?y) ∧ forNegotiation(?y, ?z) ∧ swrlb:equal(?z, "yes") → hasIssues(?x, ?y)
Rule-10	IssueStatus(?x) ∧ hasIssue(?x, ?b) ∧ hasName(?b, ?f) ∧ fromParticipant(?x, ?d) ∧ hasAgentName(?d, ?h) ∧ Strategy(?y) ∧ forIssue(?y, ?c) ∧ hasName(?c, ?g) ∧ forParticipant(?y, ?e...
Rule-11	Protocol(?x) ∧ hasBuyerNumber(?x, ?y) ∧ hasSellerNumber(?x, ?z) ∧ toIssueNumber(?x, ?a) ∧ swrlb:equal(?y, 1) ∧ swrlb:greaterThan(?z, 1) ∧ swrlb:greaterThan(?a, 1) → hasProtocolI...
Rule-2	Strategy(?x) ∧ forIssue(?x, ?d) ∧ hasName(?d, ?e) ∧ Negotiation(?y) ∧ forParticipant(?x, ?z) ∧ hasRole(?z, ?a) ∧ swrlb:equal(?a, "seller") ∧ hasIssues(?y, ?b) ∧ hasName(?b, ?f) ∧ h...
Rule-3	Strategy(?x) ∧ forIssue(?x, ?d) ∧ hasName(?d, ?e) ∧ Negotiation(?y) ∧ hasIssues(?y, ?b) ∧ hasName(?b, ?f) ∧ hasType(?b, ?c) ∧ forParticipant(?x, ?z) ∧ hasRole(?z, ?a) ∧ swrlb:e...
Rule-4	Strategy(?x) ∧ forIssue(?x, ?d) ∧ hasName(?d, ?e) ∧ Negotiation(?y) ∧ forParticipant(?x, ?z) ∧ hasRole(?z, ?a) ∧ swrlb:equal(?a, "buyer") ∧ hasIssues(?y, ?b) ∧ hasName(?b, ?f) ∧ ...
Rule-5	Strategy(?x) ∧ forIssue(?x, ?d) ∧ hasName(?d, ?e) ∧ Negotiation(?y) ∧ forParticipant(?x, ?z) ∧ hasRole(?z, ?a) ∧ swrlb:equal(?a, "buyer") ∧ hasIssues(?y, ?b) ∧ hasName(?b, ?f) ∧ ...
Rule-6	Strategy(?x) ∧ forParticipant(?x, ?y) ∧ hasPerformance(?y, ?z) → hasConcessionSpeed(?x, ?z)
Rule-7	Strategy(?x) ∧ hasConcessionSpeed(?x, ?y) ∧ swrlb:equal(?y, 1) → hasConcessionConvexity(?x, "straight")
Rule-8	Strategy(?x) ∧ hasConcessionSpeed(?x, ?y) ∧ swrlb:lessThan(?y, 1) → hasConcessionConvexity(?x, "convex")
Rule-9	Strategy(?x) ∧ hasConcessionSpeed(?x, ?y) ∧ swrlb:greaterThan(?y, 1) → hasConcessionConvexity(?x, "concave")

```
Jess Rules
(defrule Rule-9 (hasConcessionSpeed ?x ?y) (Strategy (name ?x)) (bindingFact ?f:0&:(invokeSWRLBuiltIn Rule-9 swrlb:greaterThan ?y 1)) => (assert (hasConcessionConvexity ?x "concave")) (assertOW...
(defrule Rule-8 (hasConcessionSpeed ?x ?y) (Strategy (name ?x)) (bindingFact ?f:0&:(invokeSWRLBuiltIn Rule-8 swrlb:lessThan ?y 1)) => (assert (hasConcessionConvexity ?x "convex")) (assertOWLPro...
(defrule Rule-7 (hasConcessionSpeed ?x ?y) (Strategy (name ?x)) (bindingFact ?f:0&:(invokeSWRLBuiltIn Rule-7 swrlb:equal ?y 1)) => (assert (hasConcessionConvexity ?x "straight")) (assertOWLPropert...
(defrule Rule-6 (forParticipant ?x ?y) (hasPerformance ?y ?z) (Strategy (name ?x)) => (assert (hasConcessionSpeed ?x ?z)) (assertOWLProperty "hasConcessionSpeed" ?x ?z) )
(defrule Rule-10 (hasIssue ?x ?b) (hasName ?b ?f) (fromParticipant ?x ?d) (hasAgentName ?d ?h) (forIssue ?y ?c) (hasName ?c ?g) (forParticipant ?y ?e) (hasAgentName ?e ?i) (hasFluctuation ?x ?z) (ha...
(defrule Rule-5 (forIssue ?x ?d) (hasName ?d ?e) (forParticipant ?x ?z) (hasRole ?z ?a) (hasIssues ?y ?b) (hasName ?b ?f) (hasType ?b ?c) (Strategy (name ?x)) (Negotiation (name ?y)) (bindingFact ?f:0...
(defrule Rule-11 (hasBuyerNumber ?x ?y) (hasSellerNumber ?x ?z) (toIssueNumber ?x ?a) (Protocol (name ?x)) (bindingFact ?f:0&:(invokeSWRLBuiltIn Rule-11 swrlb:equal ?y 1)) (bindingFact ?f:1&:(invoke...
(defrule Rule-4 (forIssue ?x ?d) (hasName ?d ?e) (forParticipant ?x ?z) (hasRole ?z ?a) (hasIssues ?y ?b) (hasName ?b ?f) (hasType ?b ?c) (Strategy (name ?x)) (Negotiation (name ?y)) (bindingFact ?f:0...
(defrule Rule-3 (forIssue ?x ?d) (hasName ?d ?e) (hasIssues ?y ?b) (hasName ?b ?f) (hasType ?b ?c) (forParticipant ?x ?z) (hasRole ?z ?a) (Strategy (name ?x)) (Negotiation (name ?y)) (bindingFact ?f:0...
(defrule Rule-2 (forIssue ?x ?d) (hasName ?d ?e) (forParticipant ?x ?z) (hasRole ?z ?a) (hasIssues ?y ?b) (hasName ?b ?f) (hasType ?b ?c) (Strategy (name ?x)) (Negotiation (name ?y)) (bindingFact ?f:0...
(defrule Rule-1 (forNegotiation ?y ?z) (Negotiation (name ?x)) (Issue (name ?y)) (bindingFact ?f:0&:(invokeSWRLBuiltIn Rule-1 swrlb:equal ?z "yes")) => (assert (hasIssues ?x ?y)) (assertOWLProperty "h...
```

Figure 4. The SWRLTab window in Protégé.

B. *Negotiation Interaction Test*

The principal functions of the proposed agent negotiation scheme in section 2 have been implemented in JADE. The computing engine and learning engine are built within agents to calculate the negotiation results and detect the opponent's behavior. The inference engine is operated outside the agent to infer about the negotiation strategies in this stage.

The interaction sequences have been tested in JADE. Figure 5 shows the tracing diagram of one simulation run. In this test, the one-buyer-two-sellers negotiation is converted to two individual bilateral buyer-seller negotiations through the control of a coordinator agent (CA1). CA1 sends the INFORM messages to initiate the buyer agent instances (BA1 and BA2) to carry out negotiations with seller agent SA1 and SA2 respectively. A knowledge manager agent (KMA1) is responsible for accessing the ontology knowledge base to ensure the ontology interoperability. KMA1 handles the CA1's knowledge REQUEST messages and translates the CFP and PROPOSE messages between buyer and seller agents.

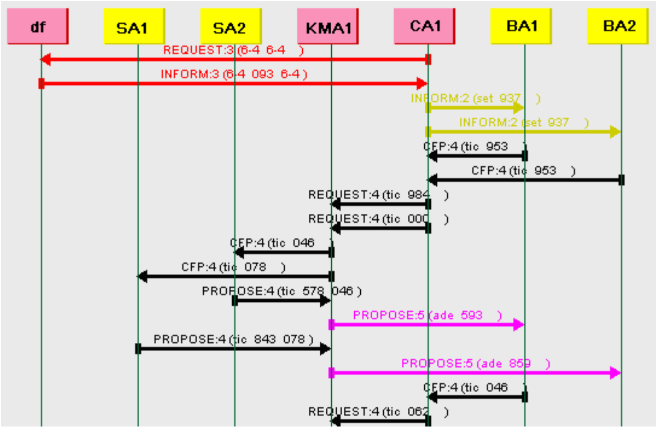


Figure 5. Agent interaction tracing diagram.

V. Conclusions and Future Work

This paper proposes an ontology-mediated approach to organize the agent-based supply chain negotiations. A generic agent negotiation scheme has been developed. It involves the agent intelligence modules, the knowledge representation method and the interaction behaviors. The negotiation ontology architecture and logic rules are defined to represent the basic concepts in the negotiation knowledge and the relationships between the concepts.

The proposed system is still in the development stage, as one direction for future work, the rules engine will be integrated into the agent platform to enable the run-time communication between inference rules and agents.

References

[1] T. Kaihara, "Multi-agent based supply chain modelling with dynamic environment " *Supply Chain Management,* vol. 85, pp. 263-269, 2003.

[2] W. S. Lo, T. P. Hong and R. Jeng, "A framework of E-SCM multi-agent systems in the fashion industry " *International Journal of Production Economics,* vol. 114, pp. 594-614, 2008.

[3] C. M. Jonker, V. Robu and J. Treur, "An agent architecture for multi-attribute negotiation using incomplete preference information " *Autonomous Agents and Multi-Agent Systems,* vol. 15, pp. 221-252, 2007.

[4] M. Wang, H. Wang, D. Vogel, K. Kumar and D. K. W. Chiu, "Agent-based negotiation and decision making for dynamic supply chain formation " *Engineering Applications of Artificial Intelligence,* vol. 22, pp. 1046-1055, 2009.

[5] R. Studer, V. R. Benjamins and D. Fensel, "Knowledge engineering, principles and methods," *Data and Knowledge Engineering,* vol. 25, pp. 161-197, 1998.

[6] V. Tamma, S. Phelps, I. Dickinson and M.Wooldridge, "Ontologies for supporting negotiation in e-commerce," *Engineering Applications of Artificial Intelligence,* vol. 18, pp. 223-236, 2005.

[7] M. C. Bravo, J. Perez, V. J. Sosa, A. Montes and G. Reyes, "Ontology support for communicating agents in negotiation processes," presented at the Fifth International Conference on Hybrid Intelligent Systems (HIS'05), 2005.

[8] A. Malucelli, D. Palzer and E. Oliveira, "Ontology-based Services to help solving the heterogeneity problem in e-commerce negotiations," *Electronic Commerce Research and Applications,* vol. 5, pp. 29-43, 2006.

[9] P. Faratin, C. Sierra and N. R. Jennings, "Negotiation decision functions for autonomous agents," *International Journal of Robotics and Autonomous Systems,* vol. 24, pp. 159-182, 1998.

[10] I. Horrocks, P. F. Patel-Schneider, S. Bechhofer and D. Tsarkov, "OWL rules: A proposal and prototype implementation," *Journal of Web Semantics,* vol. 3, pp. 23-40, 2005.