Ontology Support for Translating Negotiation Primitives

Maricela Bravo[1], Máximo López[1], Azucena Montes[1], René Santaolaya[1], Raúl Pinto[1], and Joaquín Pérez[1]
[1]Centro Nacional de Investigación y Desarrollo Tecnológico
Interior Internado Palmira S/N, Cuernavaca, Mor. 62490, México
{mari_clau, maximo, amr, rene, rpinto, jperezo}@cenidet.edu.mx,
WWW home page: http://www.cenidet.edu.mx

Abstract. In this paper we present an ontology solution to solve the problem of language heterogeneity among negotiating agents during the exchange of messages over Internet. Traditional negotiation systems have been implemented using different syntax and semantics. Our proposal offers a novel solution incorporating an ontology, which serves as a shared vocabulary of negotiation messages; and a translation module that is executed on the occurrence of a misunderstanding. We implemented a service oriented architecture for executing negotiations and conducted experiments incorporating different negotiation messages. The results of the tests show that the proposed solution improves the interoperability between heterogeneous negotiation agents.

1 Introduction

Negotiation plays a fundamental role in electronic commerce activities, allowing participants to interact and take decisions for mutual benefit. Recently there has been a growing interest in conducting negotiations over Internet, and constructing large-scale agent communities based on emergent Web service architectures. The challenge of integrating and deploying negotiation agents in open and dynamic environments is to achieve effective communications.

Traditional negotiation systems have been implemented in multi-agent systems (MAS), where agents exchange messages using an agent communication language (ACL) based on a specification like KQML [1] or FIPA [2]. These specifications provide a set of negotiation primitives based on speech act theory, and provide semantics for these primitives usage during communication. In order to facilitate effective communication, agents must be designed to be compliant with one of these

Please use the following format when citing this chapter:

Bravo, M., López, M., Montes, A., Santaolaya, R., Pinto, R., Pérez, J., 2006, in IFIP International Federation for Information Processing, Volume 217, Artificial Intelligence in Theory and Practice, ed. M. Bramer, (Boston: Springer), pp. 89–98.

ACL specifications. But the implementations of these negotiation primitives in real systems, differs in syntax and usage, because is based on proprietary program code produced by developers.

The problem of communication between negotiation agents is that even if two agents are following the same ACL, they may still suffer misunderstandings due to the different syntax and semantics of their vocabularies. In table 1, we can see that some of the reported communication languages in negotiation systems are based on FIPA, and some use a different ACL not compliant with any particular specification.

Table 1. Negotiation primitives used in different systems

Authors	ACL	Negotiation Primitives	
Jin Baek Kim, Arie Segev [7]	FIPA	Initial_offer RFQ Accept Reject Offer Counter-offer	
Stanley Y. W. Su, Chunbo Huang, Joachim Hammer [8]	FIPA	CFP Propose Accept Terminate Reject Acknowledge Modify Withdraw	
Anthony Chavez, Pattie Maes [10]	Uses a predefined set of methods, not compliant with any ACL specification.	accept-offer?(agent, from-agent, offer) what-is-price?(agent, from-agent) what-is-item?(agent, from-agent) add-sell-agent add-buy-agent add-potential-customers(sell-agent, potential-customers) add-potential-sellers(buy-agent, potential-sellers) agent-terminated(marketplace, agent) deal-made(marketplace, sell-agent, buy-agent, item, price)	
Sonia V. Rueda, Alejandro J. García, Guillermo R. Simari [11]	Based on speech act theory, not compliant with any ACL specification.	Requests_Add(s, h, p) Authorize_Add(s, h, p) Require(s, h, p) Demand(s, h, p) Accept(s, h, p)	Reject(s, h, p) Unable(s, h, p) Require-for(s, h, p, q) Insist_for(s, h, p, q) Demand_for(s, h, p, q)

Haifei Li, Chunbo Huang and Stanley Y.W Su [12]	Superset of FIPA	Call for proposal Propose proposal Reject proposal Withdraw proposal	Accept proposal Modify proposal Acknowledge message Terminate negotiation	
Jürgen Müller [6]	Based on speech act theory, not compliant with any ACL specification	Initiators: Propose, Arrange, Request, Inform, Query, Command, Inspect	Reactors: Answer, Refine, Modify, Change, Bid, Send, Reply, Refuse, Explain	Completers: Confirm, Promise, Commit, Accept, Reject, Grant, Agree.

To solve the communication problem between heterogeneous agents, we selected a translation approach based on the implementation of a shared ontology. In this ontology we explicitly describe and classify negotiation primitives in a machine interpretable form. Negotiation agents should not be forced to commit to a specific syntax. Instead, the ontology provides a shared and public vocabulary that the translator module uses to help agents to communicate during negotiation processes. We have implemented a negotiation system based on Web services technologies, into which we have incorporated the translator module and the shared ontology. Our approach acknowledges that agents may use different negotiating languages.

The rest of the document is organized as follows. In section 2, we present the translator architecture. In section 3, we describe the design of the ontology. In section 4, the general architecture of the system for executing negotiation processes is presented. In section 5, we describe the results of experiments. Finally in section 6, we present conclusions.

2 Architecture of the Translator

The translator acts as an interpreter of different negotiation agents. In figure 1, we present the architectural elements involved in translation. This architecture consists of the following elements: multiple negotiation agents, the message transport, the translator module, and the shared ontology. Each negotiation agent in turn consists of a local ACL, decision making strategies to determine the preferences, and the negotiation protocol.

For example, suppose that agents *A* and *B* initiate a negotiation process, using their own local ACL, sending messages over the message transport. If happens that agent *A* misunderstands a message from agent *B*, it invokes the semantic translator module sending the message parameters (sender, receiver, message). The translator interprets the message based on the definitions of the sender agent and converts the message into an interlingua. Then the translator converts the interlingua

representation to the target ACL based on the receiver agent definitions. Finally sends the message back to the invoking agent *A* and they continue with execution of negotiation. The translator is invoked only in the occurrence of a misunderstanding, assuring interoperability at run time.

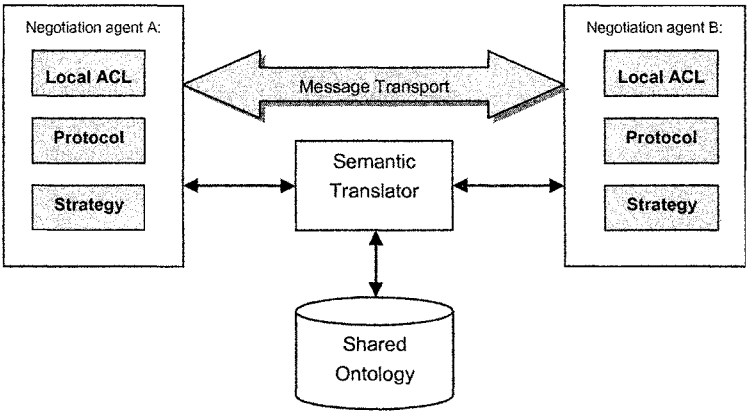

Fig. 1. Translator architecture

3 Shared Ontology

The principal objective in designing the ontology was to serve as an *interlingua* between agents during exchange of negotiation messages. According to Müller [6], negotiation messages are divided into three groups: initiators, if they initiate a negotiation, reactors, if they react on a given statement and completers, whether they complete a negotiation. We selected this classification to allow the incorporation of new negotiation primitives from the local agent ACL. Figure 2 shows the general structure of our ontology.

Based on the concepts and negotiation primitives we built our ontology. To code the ontology we decided to use OWL as the ontological language, because it is the most recent development in standard ontology languages from the World Wide Web Consortium (W3C)[1]. An OWL ontology consists of classes, properties and individuals. We developed the ontology using Protégé [14, 15], an open platform for ontology modeling and knowledge acquisition. Protégé has an OWL Plugin, which can be used to edit OWL ontologies, to access description logic reasoners, and to acquire instances of semantic markup.

[1] http://www.w3.org

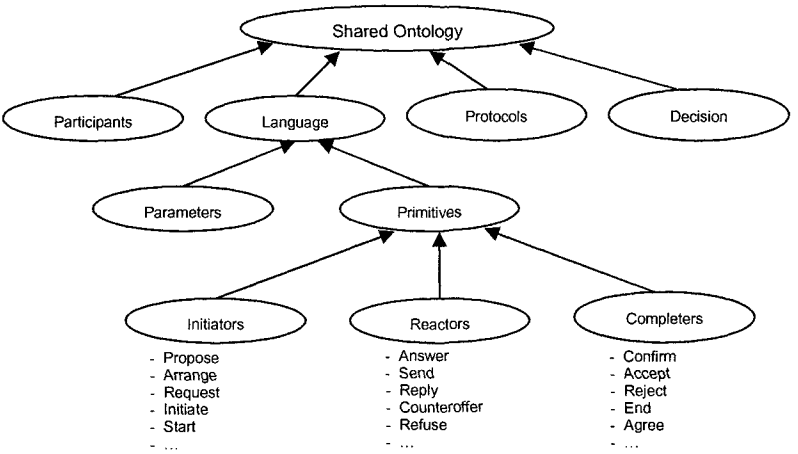

Fig. 2. General structure of the negotiation ontology

4 Implementation of the Negotiation System

The general architecture for the execution of negotiation processes is illustrated in figure 4. In this section we briefly describe the functionality and implementation techniques for each component.

a. The matchmaker is a Java module which is continuously browsing buyer registries and seller descriptions, searching for coincidences.
b. The negotiation process module is a BPEL4WS-based engine that controls the execution of negotiation processes between multiple agents according to the predefined protocols. BPEL4WS provides a language for the formal specification of business processes and business interaction protocols. The interaction with each partner occurs through Web service interfaces, and the structure of the relationship at the interface level is encapsulated in what is called a partner link.
c. Seller and buyer agents are software entities used by their respective owners to program their preferences and negotiation strategies. For example, a seller agent will be programmed to maximize his profit, establishing the lowest acceptable price and the desired price for selling. In contrast, a buyer agent is seeking to minimize his payment. On designing the negotiation agents, we identified three core elements, strategies, the set of messages and the protocol for executing the negotiation process. The requirements for these elements were specified as follows:

- Strategies should be private to each agent, because they are competing and they should not show their intentions.
- Messages should be generated privately.
- The negotiation protocol should be public or shared by all agents participating, in order to have the same set of rules for interaction. The negotiation protocol establishes the rules that agents have to follow for interaction.

d. The translator module is invoked whenever the agent misunderstands a negotiation message from another agent. The translator module was implemented using Jena2, a framework for building Semantic Web applications. It provides a programmatic environment for OWL, including a rule-based inference engine.

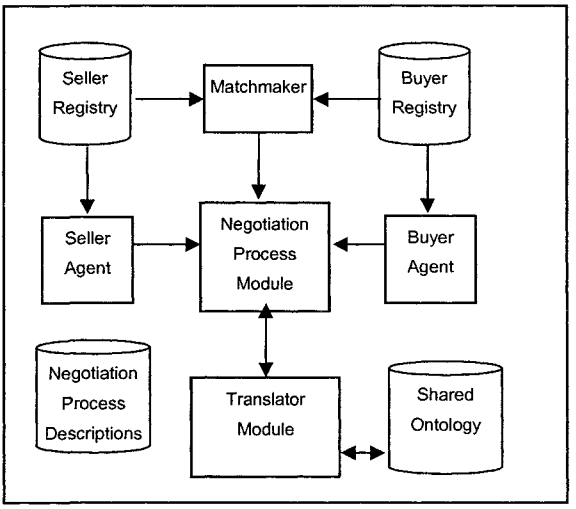

Fig. 3. Architecture of the negotiation system

5 Experimentation

In this section we describe the methodological steps that we followed for the execution of experiments.

a. Identify and describe negotiation agent's characteristics

Table 2 shows the characteristics of agents A and B, specifying their language definitions: names of primitives and a description.

[2] http://jena.sourceforge.net

Table 2. Characteristics of agents A and B

Agent A	Language definitions
	(CFP, "Initiate a negotiation process by calling for proposals"), (Propose, "Issue a proposal or a counterproposal"), (Accept, "Accept the terms specified in a proposal without further modifications"), (Terminate, "Unilaterally terminate the current negotiation process"), (Reject, "Reject the current proposal with or without an attached explanation"), (Acknowledge, "Acknowledge the receipt of a message"), (Modify, "Modify the proposal that was sent last"), (Withdraw, "Withdraw the last proposal")
Agent B	**Language definitions**
	(Initial_offer, "Send initial offer"), (RFQ, "Send request for quote"), (Accept, "Accept offer"), (Reject, "Reject offer"), (Offer, "Send offer"), (Counter-offer, "Send counter offer") (Withdraw, "Withdraw the last proposal")

b. Classify negotiation primitives in the ontology classes

For each negotiation primitive we need to analyze its semantics and usage. According to this description we can identify to which class it belongs. Table 3 shows the classification of the primitives provided by agents A and B.

Table 3. Classification of negotiation primitives

Agent	Starter	Reactor	Completer
A (Buyer)	CFP	Propose	Accept
		Modify	Reject
		Withdraw	Terminate
		Acknowledge	NotUnderstood
B (Seller)	RFQ	Initial_Offer	Accept
		Offer	Reject
		Counter_Offer	NotUnderstood

c. Align primitives in a finite state machine

Alignment is necessary to verify and clarify the intended usage of negotiation primitives.

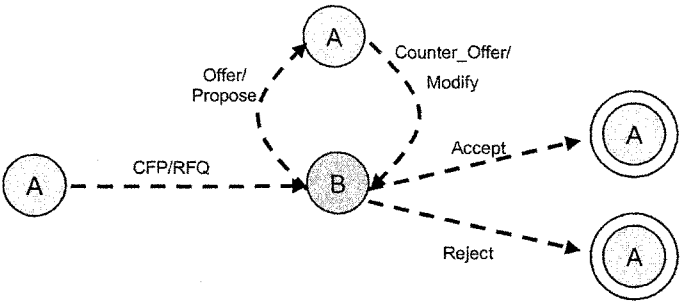

Fig. 4. Finite state machine

d. Identify and establish the relations between different primitives

Based on the classification of primitives and their allocation in the finite state machine, we can identify the relations between negotiation primitives.

A		**B**
CFP	isSynonymOf	RFQ
Propose	isSynonymOf	Offer
Propose	isSynonymOf	Inicial_Offer
Modify	isSynonymOf	Counter_Offer
Withdraw	isSynonymOf	Counter_Offer
Terminate	isSimilarOf	Reject

e. Publish and code primitives in the ontology

This step consists of populating the ontology with the primitive's definitions and relations.

f. Execute negotiation

When primitives have been published, the process of negotiation between these agents can be started. We executed 15 negotiation tests with these agents. The results of these experiments were registered in a log file. Table 4 shows the results.

Table 4. Experimental results

Test	LastPrice	MaxPay	Iterations	Quantity	FinalPrice	Result
1	$ 1,750.00	$ 1,000.00	12	1500	$ -	Reject
2	$ 774.00	$ 1,760.00	3	887	$ 1,674.00	Accept
3	$ 1,788.00	$ 128.00	12	1660	$ -	Reject
4	$ 1,058.00	$ 110.00	12	1270	$ -	Reject
5	$ 761.00	$ 77.00	2	1475	$ -	**NotUnderstood**
6	$ 621.00	$ 446.00	12	56	$ -	Reject
7	$ 114.00	$ 704.00	7	8	$ 614.00	Accept

Test	LastPrice	MaxPay	Iterations	Quantity	FinalPrice	Result
8	$ 1,837.00	$ 2,199.00	9	53	$ 2,137.00	Accept
9	$ 1,665.00	$ 2,047.00	9	56	$ 1,965.00	Accept
10	$ 1,920.00	$ 286.00	12	81	$ -	Reject
11	$ 172.00	$ 1,553.00	2	41	$ 1,172.00	Accept
12	$ 980.00	$ 1,541.00	2	67	$ -	**NotUnderstood**
13	$ 1,276.00	$ 500.00	2	43	$ -	Reject
14	$ 1,500.00	$ 1,108.00	2	110	$ -	**NotUnderstood**
15	$ 1,400.00	$ 1,520.00	3	4	$ 1,452.00	Accept

The results of experiments showed that there were some negotiations that ended the process with a *NotUnderstood* message. This was due to the emission of an *Acknowledge* message form agent A, which agent B does not recognize. Although, the experiment results show good evidence that the two agents are communicating efficiently even when their language definitions are quite different.

6 Conclusions

In this paper we have presented how an ontology approach can improve interoperability between heterogeneous negotiation agents. In particular we incorporated a translator solution for the problem of lack of understanding among seller and buyer agents during the exchange of messages at run time. We evaluated the ontology in the target application, and described the system architecture into which the negotiation processes are executed. We believe that semantic interoperability of ACL is an important issue that can be solved by incorporating a shared ontology. The experimental tests showed that the proposed architecture improves the continuity of the execution of negotiation processes, resulting in more agreements.

References

1. T. Finning, R. Fritzon, and R. McEntire: KQML as an agent communication language, in *Proceedings of the 3rd International Conference on Information and Knowledge Management*, November 1994.
2. FIPA – Foundation for Intelligent Physical Agents. FIPA Specifications, 2003; available at http://www.fipa.org/specifications/index.html.
3. Uschold, M. and King M., Towards a Methodology for Building Ontologies, *Workshop on Basic Ontological Issues in Knowledge Sharing*, 1995.
4. Grüninger, M. and Fox, M., The Role of Competency Questions in Enterprise Engineering, *IFIP WG 5.7 Workshop on Benchmarking*. Theory and Practice, Trondheim, Norway, 1994.
5. Fernández, M., Gómez-Pérez, A., and Juristo, N., METHONTOLOGY: From Onthological Art towards Ontological Engineering, *Proceedings of AAAI Spring Symposium Series*, AAAI Press, Menlo Park, Calif., pp. 33-40, 1997.

6. Müller, H. J., Negotiation Principles, *Foundations of Distributed Artificial Intelligence*, in G.M.P. O'Hare, and N.R. Jennings, New York: John Wiley & Sons.
7. Jin Baek Kim, Arie Segev, A Framework for Dynamic eBusiness Negotiation Processes, *Proceedings of IEEE Conference on E-Commerce*, New Port Beach, USA, 2003.
8. Stanley Y. W. Su, Chunbo Huang, Joachim Hammer, Yihua Huang, Haifei Li, Liu Wang, Youzhong Liu, Charnyote Pluempitiwiriyawej, Minsoo Lee and Herman Lam, An Internet-Based Negotiation Server For E-Commerce, *the VLDB Journal*, Vol. 10, No. 1, pp. 72-90, 2001.
9. Patrick C. K. Hung, WS-Negotiation: An Overview of Research Issues, *IEEE Thirty-Seventh Hawaii International Conference on System Sciences* (HICSS-37), Big Island, Hawaii, USA, January 5-8, 2004.
10. Anthony Chavez, Pattie Maes, Kasbah: An Agent Marketplace for Buying and Selling Goods, *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, London, UK, April 1996.
11. Sonia V. Rueda, Alejandro J. García, Guillermo R. Simari, Argument-based Negotiation among BDI Agents, *Computer Science & Technology*, 2(7), 2002.
12. Haifei Li, Chunbo Huang and Stanley Y.W Su, Design and Implementation of Business Objects for Automated Business Negotiations, *Group Decision and Negotiation*, Vol. 11; Part 1, pp. 23-44, 2002.
13. Dignum, Jan Dietz, Communication Modeling – The language/Action Perspective, *Proceedings of the Second International Workshop on Communication Modeling*, Computer Science Reports, Eindhoven University of Technology, 1997.
14. J. Gennari, M. Musen, R. Fergerson, W. Grosso, M. Crubézy, H. Eriksson, N. Noy, and S. Tu: The evolution of Protégé-2000: An environment for knowledge-based systems development, *International Journal of Human-Computer Studies*, 58(1): 89-123, 2003.
15. H. Knublauch: An AI tool for the real world: Knowledge modeling with Protégé, *JavaWorld*, June 20, 2003.