Group Decision and Negotiation **11**: 23–44, 2002
© 2002 *Kluwer Academic Publishers. Printed in the Netherlands*

Design and Implementation of Business Objects for Automated Business Negotiations

HAIFEI LI, CHUNBO HUANG AND STANLEY Y.W. SU
Database Systems Research and Development Center, University of Florida, Gainesville, FL 32611-6125 USA (E-mail: hli@cise.ufl.edu; chuang@cise.ufl.edu; su@cise.ufl.edu)

BENNY HIGDON
Advanced Manufacturing Solution Development, IBM, Boca Raton, FL, USA (E-mail: higdonb@us.ibm.com)

Abstract

In recent years, there has been increasing interest in the specification, generation and exchange of business objects in the context of electronic commerce. Common business objects have been defined for product catalogs, purchase orders and other business entities. However, no business objects have been defined and implemented for supporting automated business negotiations even though business negotiation is very much an integral part of business activities. In this work, we have designed and implemented a set of business negotiation objects for supporting the bargaining type of business negotiations. These objects define the operations and information contents needed for negotiation parties to express their requirements and constraints during a bargaining process. They correspond to a set of negotiation primitives, which is a superset of the negotiation-related primitives defined in two popular languages: ACL and COOL. The implementation of these objects is patterned after the business object documents in the XML format proposed by the Open Applications Group, thus conforming to the established standard. The incorporation of several types of constraint specifications in these business negotiation objects provides the negotiation parties and the negotiation servers that represent them much expressive power in specifying call-for-proposals and proposals. Two synchronization problems and their solutions associated with the withdrawal and modification of negotiation proposals are addressed and presented in this paper. The use of these business negotiation objects in a bilateral bargaining protocol is also presented. We have validated the utility of these objects in an integrated network environment, which consists of two replicated negotiation servers, two commercial products, and some other university research systems that form a supply chain.

Key words: bilateral negotiation, business object, electronic commerce, negotiation primitive, negotiation protocol

1. Introduction

Electronic commerce (Adam et al. 1999; Feldman 1998; Kalakota and Whinston 1996) has been a rapidly growing research area in recent years. Broadly speaking, electronic commerce involves conducting business activities of manufacturers, retailers, consumers, distributors, wholesalers, intermediaries, financial institutions, and service providers, electronically over a wide area computer network. At the present time, some business processes have been automated, such as enterprise resource planning, advertising, ordering, billing, payment, and accounting; however, the bargaining type of business negotiations is one of the business activities for which automation is still not available.

Negotiation (Breslin and Rubin 1991; Pruitt 1981; Raiffa 1982) is a process by which a joint decision is made by two or more parties. The negotiation parties verbalize contradictory demands at first and then move toward agreement by a process of concession making or searching for new alternatives. Negotiation is a common and important phenomenon of our everyday life. Business negotiation is the application of general negotiation principles to business settings, such as buying tangible or intangible goods, or acquiring services. Business negotiation is an important part of business activities. It can be a very time-consuming and costly one if it is not automated.

In order to conduct automated business negotiation in a wide area computer network such as the Internet, it is important to formalize the negotiation process. There are many ways to transform an unstructured negotiation process into one that is suitable for automation. An interesting one is called the proposal/counter-proposal procedure (also called offer/counter-offer procedure) as defined in (Rubinstein 1982; Ståhl 1972). Since a business negotiation usually involves only two negotiation parties, the bilateral business negotiation instead of the multi-lateral negotiation will be the focus of this paper. In a typical bilateral negotiation scenario, the first negotiation party sends a proposal to the second negotiation party, and the second negotiation party processes the proposal to determine if it is acceptable. If not, the second negotiation party can either terminate the negotiation, reject the proposal and ask the first party to modify its offer, or generate a counter-proposal, which is sent back to the first party. Proposals and counter-proposals are often exchanged between them until either a mutual agreement is reached, or one negotiation party decides to unilaterally terminate the negotiation process.

There are two important aspects of a negotiation process: communication between negotiation parties and decision-making. Communication deals with how to represent a negotiator's requirements and constraints on a product or service and how to convey his/her intent by passing messages between negotiation parties. The requirements and constraints in a proposal may list the product/service specification, price, quantity, and delivery date and their inter-relationships or constraints. Some negotiation primitive operations and a formal negotiation protocol are needed to insure meaningful message exchanges. Decision-making involves making a decision given the information provided by a negotiation partner and the internal state of a business organization (e.g., concession making, negotiation policies and strategies, and cost benefit analysis of alternative proposals). In our previous papers (Hammer et al. 2000; Su et al. 2000), we have presented the architecture, the decision-making aspects of negotiation, and the implementation of a negotiation system. A more recent version of the implemented negotiation server is described in (Su et al. 2001). In this paper, we focus on the design and implementation of business objects, which contain negotiation primitives and data that are exchanged between negotiation parties in a bilateral negotiation protocol.

Since business negotiation is viewed as an integral part of business activities, it would be ideal if message exchanges in business negotiation can be represented by a set of Business Negotiation Objects (BNOs). These objects contain negotiation primitive operations and data. They are similar to the standard business objects defined by the Open Applications Group (OAG) for other business activities, such as Request for Quotation and Process Purchase Order. In this work, we have designed and implemented a set of BNOs following the same data structures and philosophy of OAG's Business Object Documents (BODs). These Business Negotiation Object Documents (or BNODs) are expressed in XML format for transmission between automated negotiation servers.

The intended contributions of this paper are as follows. First, this paper defines the syntax and semantics of a set of negotiation primitives needed for supporting the bargaining type of business negotiations. This set of negotiation primitives is more complete than (i.e., a superset of) the set of negotiation primitives defined in FIPA's ACL (www.fipa.org/specs) and COOL (Barbuceanu 1995). Second, a couple of primitives (i.e., Modify and Withdraw) require human input and complicate the negotiation protocol because they introduce some synchronization problems in a negotiation process. Techniques for resolving these problems are presented in this paper. These two primitives and the techniques for resolving the message synchronization problems have not been dealt with in negotiation services of multi-agent systems (Rosenschein and Zlotkin 1994; Sandholm and Lesser 1995). Third, a constraint specification language is introduced in our work to enhance the expressive power of business negotiation objects. Languages used in the current NSSs like Negotiation Assistant (Rangaswamy and Shell 1997) and INSPIRE (Kersten and Noronha 1999) allow negotiation proposals to be expressed only by attribute/value pairs. Constraint specifications for dealing with more complex negotiation conditions are not supported. The negotiation objects introduced here are very general and expressive. We believe that they can be used also to enhance the negotiation efforts in the MAS and NSS areas. Fourth, the implementation of business negotiation objects (i.e., BNODs) is based on the structure and philosophy of OAGIS (www.openapplications.org), a proven industrial standard for business application integration. These BNODs have been used and validated in an operational negotiation server (Huang 2000; Su et al. 2000, 2001).

The remainer of this paper is organized as follows. Section 2 presents related work. Section 3 discusses the principles followed in our designs of BNOs. Section 4 gives an overview of the negotiation server and provides a classification of negotiation primitives. Section 5 describes the data structures of business negotiation objects. Section 6 discusses the negotiation protocol and the state transition diagrams. Section 7 shows two negotiation scenarios and the techniques used to deal with the synchronization problems associated with Modify Proposal and Withdraw Proposal. Section 8 presents the system implementation. Section 9 concludes this paper with a discussion on the lessons learned and future research.

2. Related work

FIPA's ACL has proposed several primitives for agent negotiations. However, these primitives are not adequate, in our opinion, for conducting business negotiations, in which back-and-forth bargaining is often needed. For example, FIPA's ACL does not contain primitives for the modification and the withdrawal of a negotiation proposal, which more often than not require human interventions. The COOrdination Language (COOL) allows the user to define agents, and manage communication and structured interactions between them. Like FIPA's ACL, COOL is a general language for agent communications. It is not designed specifically for addressing business negotiation problems, nor does it provide facilities to aid the negotiation decision-making process.

Müller published a survey paper on negotiation principles in the context of distributed artificial intelligence and multi-agent systems (Müller 1996). The paper lists the names of several negotiation primitives, which are divided into three groups: initiators, reactors and completers. However, the syntax and semantics of these primitives are not described. Also, no concrete negotiation protocol(s) nor negotiation scenario(s) are provided to show the use of these primitives.

Bui, Bodart and Ma proposed a formal language called ARBAS to support argumentations in network-based organizations (Bui et al. 1997). ARBAS is formally defined at three levels. At the first level, a *lexicon* is defined to provide a vocabulary for the argumentation discourse. At the second level, a *syntax* that consists of a simple set of rules is used to compose an argument. At the third level, *Semantic Constraints* defined in terms of rules are used to guarantee the semantic correctness of an argument. The language is based on the theoretical framework presented by Jackson and Jacobs (Jackson and Jacobs 1980), who proposed a theory of argument.

Chang and Woo proposed a negotiation protocol called Speech-Act-based Negotiation Protocol (SANP) (Chang and Woo 1994). SANP is based on Ballmer and Brennenstuhl's speech act classification and on the negotiation analysis literature. It is a communication-level protocol. Like other related works, the protocol is represented as a state transition diagram. A sentence (similar to our concept of BNO) in SANP consists of two parts: <function> <content>, where <function> is a speech act category name, and <content> is the representation of domain knowledge. The system is implemented based on an electronic conversation toolkit called Strudel (Shepherd et al. 1990). Representation of domain-dependent knowledge and the reasoning mechanism used by individual agents are not discussed in the paper. Unlike SANP, knowledge representation is addressed in our designs of BNOs. Although our implemented system has reasoning and decision-making components, they are out of the scope of this paper. Interested readers are encouraged to look up the references to the implemented negotiation server given previously.

Conklin and Begeman proposed a Graphical Issue Based Information System (gIBIS) (Conklin and Begeman 1988). It is an application-specific hypertext system designed to facilitate the capture of early design deliberations. It is based on the IBIS method developed by Horst Rittle (Kunz and Rittel 1970). gIBIS is an implementation of IBIS in a graphical form. The IBIS method focuses on the definition and manipulation of IPA (Issue-Position-Argument) nodes. Each *issue* can have many *positions*. A *position* is a statement or assertion that resolves the *issue*. Each *position* may have one or more *arguments* that either support the *position* or object to it. gIBIS is a useful tool for collaboration in the early design stage of various design situations, such as architectural design, city planning, and planning at the World Health Organization. It is not clear whether it is applicable to the business negotiation environment. In business negotiation, negotiation parties do not necessarily collaborate to reach an agreement. They often compete with each other and hide the confidential information from each other to maximize their own benefits.

Negotiation Support System (NSS) (Lim and Benbasat 1993) is one type of system designed to aid human decision-making in a negotiation process. NSSs, such as Negotiation Assistant and INSPIRE, use a relatively simple language to represent negotiation messages. Both systems use attribute/value pairs to represent proposals (or offers). One example of an offer in INSPIRE is “price = \$3.47, Delivery = 20 days, Payment = 30 days after delivery, returns = 75% refunds with 10% spoilage”. They do not provide the language facility to express complex attribute and inter-attribute constraints. Proposals in NSSs are used purely for human negotiators to make decisions.

3. Design principles

The principles that guide our designs of BNOs and the system architecture for supporting automated business negotiation are as follows.

First, the designs of BNOs should mimic the language used in human-based negotiations. Similar to the work of SANP, our designs of BNOs are based on the theory of speech act (Austin 1962; Searle 1969, 1975). They correspond to the speech acts such as propose, accept, reject, withdraw, and modify a negotiation proposal, acknowledge the receipt of a proposal, and terminate a negotiation process. We believe that the above verbs are essential for both human-based negotiations and automated negotiations. Needless to say, there are many other verbs used in human speech communications. They have been adopted by the agent communication language ACL; however, they are not directly relevant to business negotiations, in our opinion.

Second, the designs of BNOs must be expressive and powerful enough to allow the specification of, not only a user/organization’s requirements, but also the constraints associated with a product or service. In our opinion, the attribute-value pair type of product/service specification supported by most negotiation support systems is too restrictive and does not reflect the real world negotiation situations in which constraints on products and services are often exchanged between human negotiators. In our work, negotiation proposals specified in XML BNODs can have rather complex structures and constraints (i.e., attribute and inter-attribute constraints). Also, multiple product or service items and their inter-item constraints can be specified in a single negotiation proposal. This is useful because a company may want, for example, to buy fifty bicycles and fifty bicycle helmets, only if the bicycle shop can sell the same quantity of these two items and deliver them in three days.

Third, the designs of BNOs should allow the specification of alternative values associated with a product attribute. For example, a buyer may not be exactly sure about what he/she wants for a computer purchase and how much he/she wants to pay. He/she may specify in a proposal that the monitor size can either be 15, 17, 19 or 21 inches and the price should be in the range between \$900 and \$1,500. In our work, we allow the use of enumeration and value range to specify attribute constraints associated with a product or service. In a negotiation proposal, if several attributes associated with a product or service have multiple values, different value combinations would form the specifications of different alternative products or services. They need to be evaluated to determine their costs and benefits to the receiver of the proposal so that a proper value combination (i.e., a specific product or service) can be selected. In our work, a constraint satisfaction processing server and a cost-benefit evaluation server are used to aid this decision-making.

Fourth, the system architecture designed for supporting Internet-based business negotiations must be scalable. It can be expected that, if the automated negotiation service is available on the Internet, a large number of Internet users and business organizations will be interested in using the service. A centralized negotiation server will not be able to sustain the performance if the number of users/organizations continues to increase. In our work, the system architecture is a peer-to-peer architecture. The negotiation server we have developed can be replicated and installed at many Web sites to provide negotiation services. Each negotiation server can have many clients (users or organizations) who register with the server. The server will conduct automated negotiations with other servers on behalf of its clients, based on their registered requirements, constraints, preferences, etc. Being a server, it can carry out multiple negotiation transactions concurrently.

Fifth, the system architecture and the designs of BNOs should consider the issue of trust. Human-based business negotiations are usually conducted without third party intervention and mediation. In a typical human-based negotiation case, a purchase manager of one company would directly bargain with a sales manager of another company about the terms and conditions of a business deal. No third party is involved. One possible reason is that both parties are usually not willing to share confidential information, such as negotiation strategies, product preferences, and cost-benefit evaluation methods/formula, with others. Theoretically, third party mediation is ideal and can greatly speed up the negotiation process because it knows the requirements and constraints of both negotiation parties, and can thus suggest alternatives that meet the requirements and constraints of both sides or point out the best solution for both. However, in practice, few business negotiations are conducted through a third-party mediation because of the lack of trust. In our system architecture, an organization can install its own negotiation server on its own computer. The server will manage the organization's requirements, constraints, negotiation strategies, preferences, and other confidential information and conduct automated negotiations on its behalf. The organization can have full and secured control over the server and its information contents.

The above design principles and requirements have guided our designs of BNOs, BNODs, and the system architecture to be presented in this paper.

4. Negotiation server and negotiation primitives

4.1. System architecture of the negotiation server

In order to give readers an overall picture of how BNOs are used in a Negotiation Server (NS), we first give an overview of the system architecture of the server (Figure 1). The Negotiation Server is replicated and attached to Web servers on the Internet. These multiple copies of NS provide negotiation services to clients (persons and/or organizations) who have registered with them. They conduct negotiation processes with each other on behalf of their clients, based on the pre-registered requirements, constraints, preferences, and negotiation strategies. Multiple negotiations can thus be carried out concurrently and in a distributed fashion. The approach used in this work is different from the one used in a

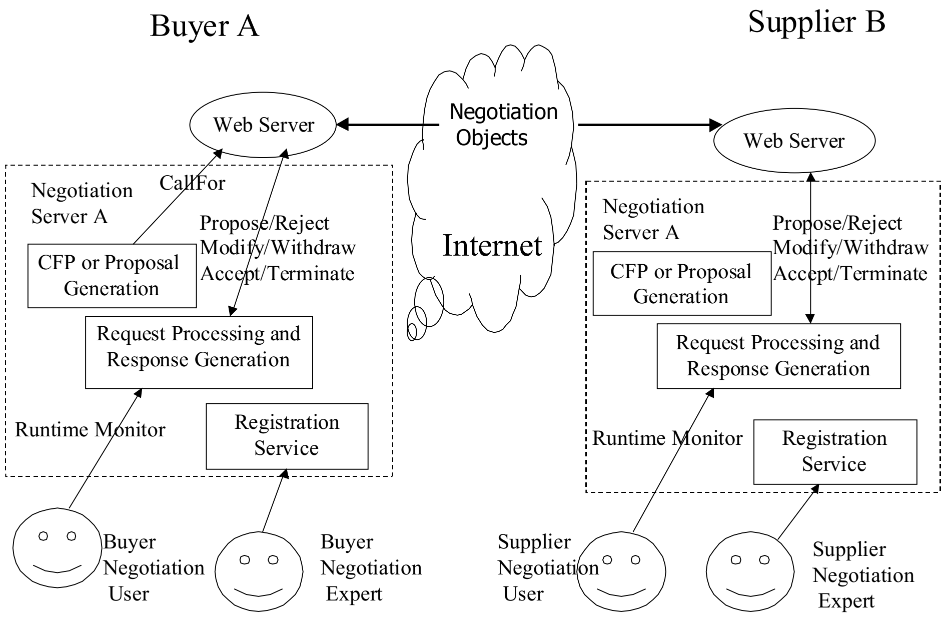
Buyer A
Supplier B
Web Server
Negotiation Objects
Internet
Web Server
Negotiation Server A
CallFor
Propose/Reject Modify/Withdraw Accept/Terminate
CFP or Proposal Generation
Request Processing and Response Generation
Runtime Monitor
Registration Service
Buyer Negotiation User
Buyer Negotiation Expert
Negotiation Server A
Propose/Reject Modify/Withdraw Accept/Terminate
CFP or Proposal Generation
Request Processing and Response Generation
Runtime Monitor
Registration Service
Supplier Negotiation User
Supplier Negotiation Expert

Figure 1. System architecture.

centralized electronic marketplace (Collins et al. 1998; Lee et al. 1997; Myerson and Satterthwaite 1983; Reich and Ben-Shaul 1998).

Figure 1 shows the key build-time and run-time components of the server. The left side of the figure is a buyer negotiation server A, and the right side is a supplier negotiation server B. These servers are identical and are attached to Web Servers to make use of the Web services. The bubble in the middle is the Internet through which BNODs are exchanged. The three major components of the negotiation server are shown inside the dotted boxes. Their functions are Call-for-Proposal or Proposal Generation, Proposal Processing and Response Generation, and Registration Service, respectively.

At build-time, negotiation experts use the Registration Service component to register their requirements, constraints, preferences, and negotiation strategies (in the form of rules) that are associated with the products/services they want to sell/provide or acquire. Negotiation strategic rules can be modified dynamically (i.e., at run-time). At run-time, the CFP/ Proposal Generation component of the buyer's negotiation server generates either a CFP or a Proposal (if the buyer knows exactly what he/she wants to buy), which is sent to the supplier to start a negotiation transaction. The CFP or Proposal is analyzed by the Proposal Processing and Response Generation component to produce a response to the call-for-proposal or the proposed purchase. All messages exchanged are specified in XML BNODs. Proposals and counter-proposals can go back and forth between two negotiation servers many times until either both sides reach an agreement or one side decides to unilaterally terminate the negotiation process. The negotiation user of both sides can monitor the process and intervene whenever necessary by changing the contents of BNODs and/or negotiation strategic rules.

4.2. Negotiation primitives

Eight primitive operations for negotiation are introduced in our work. They are: CallFor Proposal (CFP), Propose Proposal, Reject Proposal, Withdraw Proposal, Accept Proposal, Modify Proposal, Acknowledge Message, and Terminate Negotiation. They are specified in the form of a verb and a noun to conform to the business object specification of OAG. Table 1 gives a brief description for each primitive.

CallFor Proposal and Process Proposal are two alternative ways of initiating a negotiation transaction. Accept Proposal and Terminate Negotiation are two alternative terminators that can bring a negotiation transaction to an end. Acknowledge Message is used to acknowledge the receipt of a BNOD. The remaining primitives together with Process Proposal are iterators, which can be used repeatedly in a negotiation process to do bargaining. Since a negotiation party is allowed to proactively propose an offer to the other side, Propose Proposal is used as both an initiator and an iterator. The situation is similar to the one where an “unsolicited” proposal is presented to a buyer from a supplier or vice versa. The conceptual state transition using these negotiation primitives is relatively simple. Basically, a negotiation transaction is an “initiation-iteration-termination” procedure. Section 6 will describe in detail the state transition diagrams that define the negotiation protocol.

5. Business negotiation objects

As shown in Section 4, each negotiation primitive is composed of a verb and a noun. The verb names the action of the sender and the noun names the information contents to be processed by the receiver. BNOs are the formal specifications of these primitives. In this

Table 1. Negotiation primitives

Negotiation primitive	Brief description
Call for proposal	Initiate a call-for-proposal
Propose proposal	Send a proposal or a counter-proposal, which specifies the constraints and conditions acceptable to the sender
Reject proposal	Reject the received proposal with or without an attached explanation and expect the negotiation partner to modify and re-send the modified proposal
Withdraw proposal	Withdraw the previous proposal that was sent
Accept proposal	Accept the terms and conditions specified in a proposal without further modifications
Modify proposal	Modify the CFP or the proposal that was sent
Acknowledge message	Acknowledge the receipt of a BNOD
Terminate negotiation	Unilaterally terminate the negotiation process

section, we focus on the information contents that go with the verbs: CALLFOR, PROPOSE, ACCEPT, MODIFY, TERMINATE, WITHDRAW, REJECT and ACKNOWLEGE. We distinguish two general types of information contents in the formal specification: attributes and constraints.

5.1. Attributes

Attributes are used to characterize a negotiation item (or items) specified in a proposal as well as the conditions and terms associated with a business transaction. For example, in a computer purchase, the negotiation item can be characterized by monitor type, memory size, CPU speed, etc. Other attributes are used to characterize the business transaction that involves the item being negotiated. Price, quantity, delivery date, return policy, and terms and conditions are attributes associated with the transaction. We can therefore distinguish item attributes from transactional attributes. Although, during a negotiation process, negotiation parties may bargain for both types of attributes, it is useful to separate these two general types from the data modeling point of view. Therefore, in our design of BNOs, these two types of attributes are specified in different sub-structures.

5.2. Constraints

Constraints (Marriott and Stuckey 1998; Tsang 1993) associated with negotiated items can be specified in a negotiation proposal. Three general types of constraints are distinguished: attribute constraints, inter-attribute constraints and inter-item constraints. An attribute constraint specifies the constraint on the value of an attribute. For example, the memory size is greater than 64M bytes, or equal to one of the enumerated values, or falls in a specified value range. An inter-attribute constraint specifies the relationship among a number of attributes and values. For example, a constraint may state that "if the model of the computer is Pentium II 500 and the disk capacity is greater than 10 GB, then the memory size has to be greater than 64MB". Inter-item constraints are used to specify the relationship among multiple negotiation items specified in a negotiation proposal. An example would be "if I can not get the same number of Ethernet cards to go with the number of computers I am ordering, I do not want both of these items," meaning the quantities of these two items have to be the same. In our designs of BNOs, attribute constraints are specified together with attributes, and inter-attribute constraints are specified at the level of a data structure above the specification of both item attributes and transactional attributes because the constraints may involve both types of attributes and values. Inter-item constraints are specified at the level above the specification of negotiation items for obvious reasons.

The above three general types of constraints can be specified in different constraint languages with different syntaxes. Different application systems and users may have their own preferences on the languages they use. For that reason, it is important to leave the languages open. In the designs of negotiation objects, constraint specifications are values of some general attributes. Thus, the interpretation of the constraint specifications is left to the application system that receives a BNOD. A new field called ConstraintLanguage, whose value names a constraint specification language, is added to a common header of a BNOD. UFCL (Huang 2000) is the constraint specification language used in our implementation.

5.3. Design of business negotiation objects

The following notation similar to EBNF (Extended Backus-Naur Form) and XML DTD is used to describe the structures of BNOs: Negotiation_Object_Name (Substructure1, Substructure2+, Substructure3*, substructure4? . . .). Substructures are defined in terms of their own underlying structures. The process repeats until the substructure is of string type or numeric type. Symbols like +, * and ? at the end represent one or many, zero or many, zero or one (optional) occurrences, respectively.

All NBOs share some common characteristics. Therefore, it is convenient to group these characteristics into a common data structure called HEADER. The structure of HEADER is:

HEADER (NEGTRANSACTID, TRANSACTDESCRIPTN, TRANSACTNAME, SENDERNS, SENDERUID, RECEIVERNS, RECEIVERUID, PROTOCOL, PROPID, PROPDESCRIPTN, CONSTRAINTLANGUAGE)

A negotiation transaction refers to the whole negotiation process from the sending (or receiving) of an initiator object to the sending (or receiving) of a terminator object. There are three fields to describe transaction information. NEGTRANSACTID is the unique negotiation transaction identifier that is automatically generated by the system. TRANSACTNAME is the negotiation transaction name given by the user. TRANSACTDESCRIPTN is the textual description of the negotiation transaction.

There are four fields to identify negotiation servers and negotiation users. Since a negotiation server may serve multiple users, it is important to include identifiers for both the negotiation server and its users. SENDERNS is the sender negotiation server identifier. SENDERUID is the sender user identifier. RECEIVERNS is the receiver negotiation server identifier. RECEIVERUID is the receiver user identifier.

PROTOCOL is the protocol used. “Bargaining” is the protocol used in our current implementation of the negotiation server. The future extension may include other protocols, such as “bidding” and “auction”. PROPID is the unique proposal identifier. Among eight BNOs, only CallFor Proposal and Propose Proposal generate new proposal identifiers. PROPDESCRIPTN is the textual description of the proposal. CONSTRAINTLANGUAGE is the name of a constraint specification language discussed previously.

The BNO for CallFor Proposal is generally used by a buyer to ask suppliers to send in sale proposals; however, it can also be used by a supplier to call for buyers’ proposals to buy an advertised product or service. This negotiation object may or may not contain the actual requirements and constraints of the buyer/supplier for some specified items besides their names. If the actual requirements are specified in this object, the object has the same functionality as the BNO for Propose Proposal. The following is the structure:

CALLFOR_PROPOSAL (HEADER, NEGITEM*, INTERITEMCONSTR*)
NEGITEM (ITEMID, ACTIONTYPE, DESCRIPTN?, TRANSACATTR*, ITEMATTR*, INTERATTRCONSTR*)
TRANSACATTR (ATTRNAME, ATTRTYPE, CONSTRAINT?)
ITEMATTR (ATTRNAME, ATTRTYPE, CONSTRAINT?)
INTERATTRCONSTR (IACNAME, CONSTRAINT?)
INTERITEMCONSTR (IICNAME, CONSTRAINT?)

The BNO for CallFor Proposal has a HEADER, zero to many NEGITEM (negotiation item) and zero to many INTERITEMCONSTR (inter-item constraint). NEGITEM has zero to many TRANSACATTR (transactional attribute), ITEMATTR (item attribute), and INTERATTRCONSTR (inter-attribute constraint). ITEM refers to either tangible or intangible goods or intangible services. ITEMID is the identifier for the item. ACTIONTYPE indicates the action to be taken on the item. It has a value of "buy", "sell", "lease", etc. DESCRIPTN is the textual description of the item. ATTRNAME is the attribute name and ATTRTYPE is the attribute type. CONSTRAINT is the constraints posed in the constraint specification language specified in the HEADER.

The BNO for Propose Proposal sends a proposal or a counter-proposal, which specifies the constraints and conditions acceptable to the sender. It implicitly asks the receiver to evaluate the proposal contained in the BNO. The BNO for Propose Proposal has the same structure as the one for CallFor Proposal.

The BNO for Accept Proposal accepts the terms and conditions specified in a proposal without further modifications. It contains the specific description of item(s) that have been accepted.

ACCEPT_PROPOSAL (HEADER, PROP)
PROP (PID, ACCEPTEDITEM+)
ACCEPTEDITEM (ITEMID, DESCRIPTN?, BUSINESSATTR*, ITEMATTR*)

PROP (stands for PROPosal) is the structure for specifying the item(s) to be accepted. It includes a PID (proposal identifier) and at least one ACCEPTEDITEM (accepted item). ACCEPTEDITEM includes zero to many TRANSACATTR and zero to many ITEMATTR. Inter-attribute constraints and inter-item constraints are missing from the definition. The reason is that these constraints should have been satisfied when an agreement has been reached.

The BNO for Terminate Negotiation tells the negotiation partner that the negotiation transaction has been unilaterally terminated. The sender refuses to conduct further negotiation on the item(s). The BNO for Terminate Negotiation is a simple object that contains a HEADER and NEGTRANSACT. NEGTRANSACT is a structure that includes the identifier of the negotiation transaction that is to be terminated. The following is the structure:

TERMINATE_NEGOTIATION (HEADER, NEGTRANSACT)
NEGTRANSACT (NEGTRANSACTID)

The BNO for Reject Proposal tells the negotiation partner that the previous proposal is not acceptable; however, the sender does not want to terminate the negotiation and is willing to accept a modified proposal from the receiver. This negotiation object may optionally give the reason for rejection by pointing out the attributes and constraints in the received proposal that are not acceptable. This object does not terminate the negotiation process. Rather, it is used to ask the other negotiation party to modify its proposal by eliminating the unacceptable conditions. The following is the structure:

```
REJECT_PROPOSAL (HEADER, REJ)
REJ (PID, REJREASON+)
REJREASON (ITEMID, VIOLATEDCONSTRNAME?, REASON)
```

REJ (stands for REJect) is the structure for rejection. REJ has one PID (proposal ID) and one to many REJREASON. REJREASON includes an ITEMID (item identifier) and the name of the violated constraint and a textual explanation of the reason.

The BNO for Modify Proposal tells the negotiation partner that it wants to modify the last proposal that was sent. This object specifies the modifications that the sender wants to make. It allows the sender to update, insert, and delete the attributes (and their constraints), inter-attribute constraints, and inter-item constraints. It is semantically equivalent to withdraw the last proposal and send a new proposal to the negotiation partner; however, it is useful to have a separate object for this purpose when the sender wants to make a minor change to a complex proposal. ITEMMOD stands for item modification, IAC stands for Inter-Attribute Constraint, and IIC stands for Inter-Item Constraint in the following description. The following is the structure:

```
MODIFY_PROPOSAL (HEADER, PID, ITEMMOD*, IIC_UPDATE*, IIC_INSERT*,
IIC_DELETE*)
ITEMMOD (ATTR_UPDATE*, ATTR_INSERT*, ATTR_DELETE*,
IAC_UPDATE*, IAC_INSERT*, IAC_DELETE*)
ATTR_UPDATE (ATTRNAME, ATTRTYPE, CONSTRAINT?)
ATTR_INSERT (ATTRNAME, ATTRTYPE, CONSTRAINT?)
ATTR_DELETE (ATTRNAME)
IAC_UPDATE (IACNAME, CONSTRAINT?)
IAC_INSERT (IACNAME, CONSTRAINT?)
IAC_DELETE (IACNAME)
IIC_UPDATE (IICNAME, CONSTRAINT?)
IIC_INSERT (IICNAME, CONSTRAINT?)
IIC_DELETE (IICNAME)
```

The design of the BNO for Modify Proposal is patterned after three common modification operations: Update, Insert and Delete. There are three levels of modifications: attribute level, inter-attribute level and inter-item level. Update and Insert have the same structure, but Delete has only one field. The reason is that it is enough to specify the data item name to be deleted. ATTR_UPDATE and ATTR_INSERT have an additional field called ATTRTYPE.

The assumption is that it is possible to change the attribute type; e.g., from an integer to a string.

The BNO for Withdraw Proposal tells the negotiation partner that it wants to withdraw the CFP or Proposal that was sent last. When the receiver receives this object, it needs to discard the previously received proposal and wait for a further message from the sender. After sending this object, the sender will generally send another message to the negotiation partner for further processing. The following is the simple structure:

WITHDRAW_PROPOSAL (HEADER, PID)

The BNO for Acknowledge Message tells the negotiation partner that a message has been received. This message is optional for most negotiation primitives except the Propose and CallFor messages. The acknowledgement for Propose and CallFor is used to deal with a couple of synchronization problems to be addressed in Section 7. This BNO has only a header data structure as shown below:

ACKNOWLEDGE_MESSAGE (HEADER)

5.4. Implementation of business negotiation objects based on OAGIS

All the BNOs described above are implemented following the structure and format of XML BODs (Business Object Documents in XML syntax) proposed by OAG. There is a DTD for each negotiation object. HEADER is implemented using a common data type called NEGBODHEADER. Action names such as Propose, Reject, etc., are specified as the VERBs of BODs, and the data item names are mapped to the NOUNs. Structures such as NEGITEM, and PROP are mapped to corresponding data types. What we called business negotiation object documents (BNODs) are the XML-wrapped BNOs.

6. Negotiation protocol and state transition diagram

To ensure an effective and meaningful communication between negotiation servers, they must follow a well-defined protocol to exchange negotiation objects in a negotiation process. For instance, after the buyer's negotiation server sends out a CFP, it expects to receive a Propose Proposal from the negotiation partner. Other types of messages would not be meaningful. Several negotiation protocols for bidding and different types of auctions are available and have been used in some automated systems. In this paper, we use two state transition diagrams to define a bilateral bargaining protocol. Figure 2 defines the states and state transitions of the buyer, and Figure 3 defines those of the supplier.

There are a total of 15 different negotiation states that a negotiation server can be in during a bilateral bargaining process. Since a negotiation server can be the initiator of a negotiation transaction and, at the same time, the negotiation partner of a transaction initiated by another server, the transition diagrams define the various states that a server can be in when

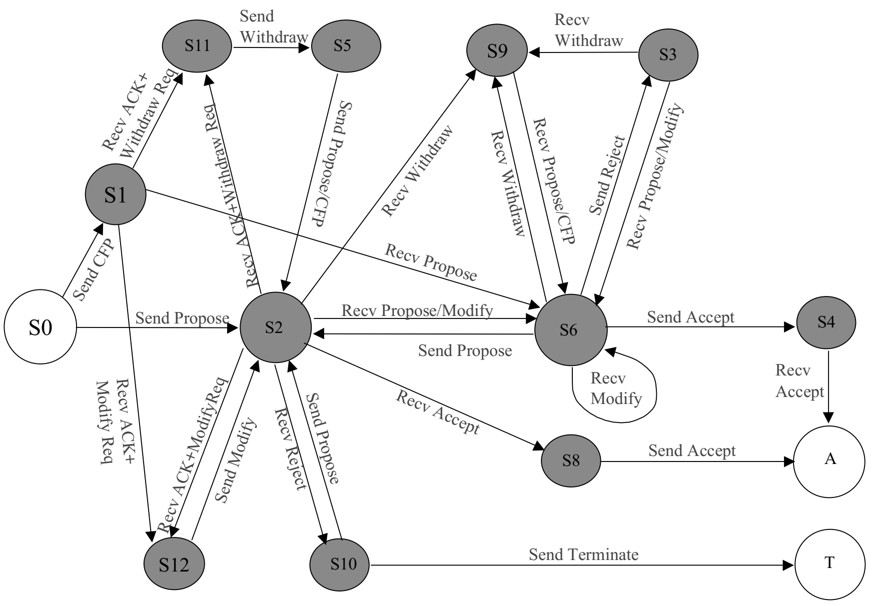

Figure 2. Buyer's negotiation protocol and state transition diagram.

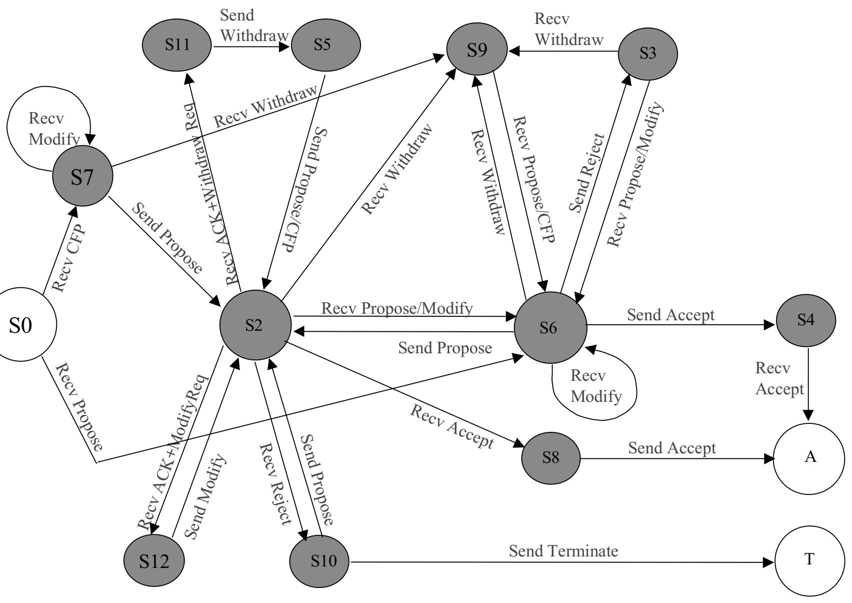

Figure 3. Supplier's negotiation protocol and state transition diagram.

playing both roles. We shall use “buyer” and “supplier” to represent these two roles. The meanings of these states are shown in Table 2.

To simplify the explanation of Table 2, we shall use the verbs of negotiation primitives instead of their corresponding BNODs, which are actually transmitted between negotiation servers. S0 is the initial state. S1 is entered after the buyer sends out a CallFor. S7 is entered when the supplier receives a CallFor. The remaining states are shared by both the buyer and the supplier. In general, a negotiation transaction is initiated by the buyer after sending out a CallFor. In the CallFor, the buyer indicates its interest to start a negotiation on some negotiation item(s) and provide some specification and constraints associated with the item(s). To respond to a CallFor, the supplier would send a Propose (i.e., proposal) and provide the negotiation conditions and constraints from its own perspective. Then, the buyer can decide whether the constraints in the proposal are acceptable or not. If they are not acceptable, further negotiation needs to be conducted to resolve the differences by sending a Propose (i.e., counter-proposal) to the supplier. In another case, if the buyer knows precisely the constraints of a negotiation item it wants to send to a supplier, it can use Propose to start the negotiation instead of CallFor. Hence, the supplier may also receive a Propose to start a negotiation. Either case (CallFor or Propose) is defined by the messages and state transitions surrounding S0.

The most frequently used primitive is Propose, which transmits a proposal or counter-proposal between two negotiation servers. S2 and S6 are two states of a negotiation server when it sends and receives Propose, respectively. To respond to a Propose, several messages are allowed: another Propose to issue a counter-proposal, a Reject to indicate the rejection of some conditions specified in the original proposal, and an Accept to indicate the acceptance of the terms and conditions specified in the previous proposal without further modifications. Note that in our negotiation protocol, a Reject message is not for specifying the termination of the negotiation. Rather, it is used to convey the dissatisfaction of some conditions to the negotiation partner, and for the partner to modify and re-send a

Table 2. Negotiation state semantics

S0	Initial state
S1	CFP is sent
S2	Propose is sent
S3	Reject is sent
S4	Accept is sent
S5	Withdraw is sent
S6	Propose/CFP/Modify is received
S7	CFP/Modify is received
S8	Accept is received
S9	Withdraw is received
S10	Reject is received
S11	Received the Acknowledge of the Propose/CFP that was sent and the user requests the Withdraw of that Propose/CFP
S12	Received the Acknowledge of the Propose/CFP that was sent and the user requests the Modify of that Propose/CFP
A	Agreement is reached
T	Negotiation is unilaterally terminated

proposal back. Thus, the Reject contains the rejected conditions. For a negotiation server to unilaterally terminate a negotiation process for any reason, the Terminate can be used. The state transitions and messages between state S2, S10, and T describe the cases of rejection and termination. In fact, at any state except the initial state S0, if a Terminate is received or sent, the negotiation state would transit to T and the negotiation process terminates. However, in order not to clutter the diagram, the state transitions to T are not shown.

To reach an agreement in a bilateral negotiation, both sides exchange a pair of Accept messages containing identical conditions that made the deal. This is the same as putting both signatures on an agreed contract. It allows both sides to ensure that their counterparts have accepted the conditions as specified. The messages from S2 to S8, and from S8 to state A illustrate this case. A suggested alternative is for the receiver of an Accept message to simply return an acknowledgement instead of the second Accept. However, this may lead to a misunderstood agreement if the contents of the first Accept has been altered or corrupted during its transmission.

7. Negotiation scenarios

To further clarify the semantics and usage of BNOs and the negotiation protocol, a few negotiation scenarios are used to show the meaningful state transitions and message exchanges between two negotiation servers.

(1) A counter-proposal scenario

Figure 4 shows that a buyer sends a CallFor to a supplier and the supplier sends a Propose back. The buyer is not satisfied with the conditions, so the buyer sends a Reject to the supplier. After the supplier receives a Reject, the supplier returns a modified proposal (Propose) back to the buyer (Buyer State: S0->S1->S6->S3->S6, Supplier State: S0->S7->S2->S10->S2). This time, the buyer is willing to accept the modified proposal (Buyer State: S6->S4->A, Supplier State: S2->8->A). The supplier confirms the acceptance from the buyer by sending an Accept to the buyer.

(2) A scenario with Modify/Withdraw message exchange

Sometimes a negotiator wants to change some conditions in the previous proposal or withdraw the previous proposal that has been sent. Two BNOs are provided for the two primitives: Modify and Withdraw. When a negotiation server receives a Modify or Withdraw message, it should stop the processing of the previous Propose message. In the case of Modify, the negotiation server needs to combine the contents of the previous proposal with the modifications specified in the BNOD for Modify to produce a new proposal. In the case of Withdraw, the negotiation server will receive a new proposal from the sender of Withdraw. Both cases require the negotiation server to respond based on the new proposal. As

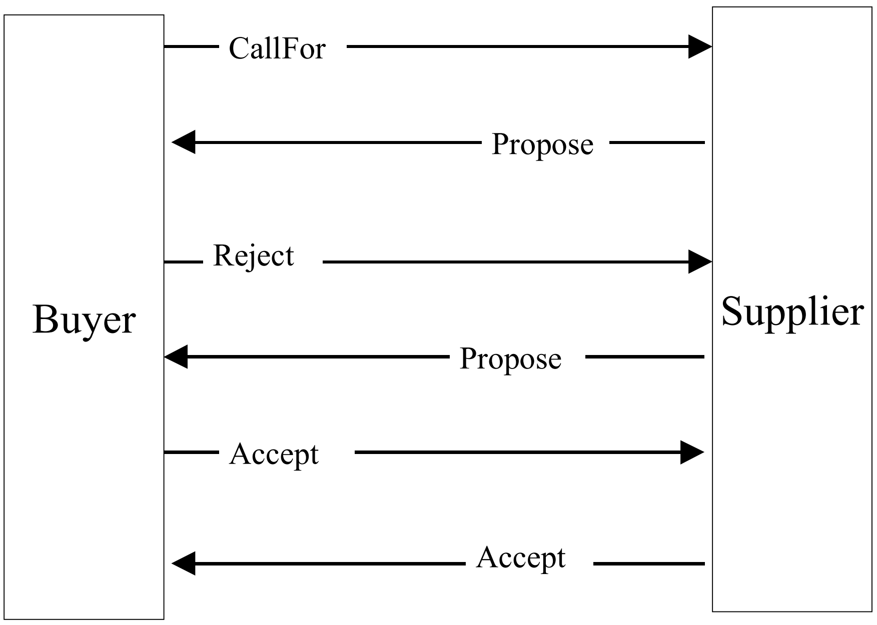

Figure 4. Counter-proposal scenario.

described in the state transition diagram, Modify and Withdraw for the CallFor message are the same as those for Propose. Moreover, since the modification or withdrawal of a Propose or a CallFor generally requires some human intervention, a human input action, ModifyReq (for modification request) or WithdrawReq (for withdraw request), is introduced in the protocol state diagrams. A scenario that involves Modify/Withdraw is shown in Figure 5.

In this scenario, the buyer sends out a CallFor to the supplier (Buyer State: S0->S1, Supplier State: S0->S7). Before the supplier returns any message, the buyer changes his/her mind and uses the Modify to notify the supplier of several changes (Buyer State: S1->S12->S2, Supplier State: S7->S7). The supplier then uses the new information to generate a proposal and sends it back to the buyer (Supplier State: S7->S2, Buyer State: S2->S6). Before the buyer responds, the supplier also changes his/her mind. The supplier then uses the Withdraw to first notify the buyer of his/her intention and then sends a new Propose to the buyer (Supplier State: S2->S11->S5->S2, Buyer State: S6->S9->S6). The buyer rejects the new proposal (Reject) and the supplier sends another counter-proposal (Propose) (Buyer State: S6->S3->S6, Supplier State: S2->S10->S2). The buyer terminates the negotiation without reaching an agreement. (Buyer State: S6->T, Supplier State: S2->T).

The last scenario is an uncomplicated case of using Modify/Withdraw in a negotiation. Unlike other negotiation primitives which are designed to respond to messages received from the negotiation partner, Modify and Withdraw primitives are designed to change the proposal that was sent out by a negotiation server. Sometimes this may cause some syn-

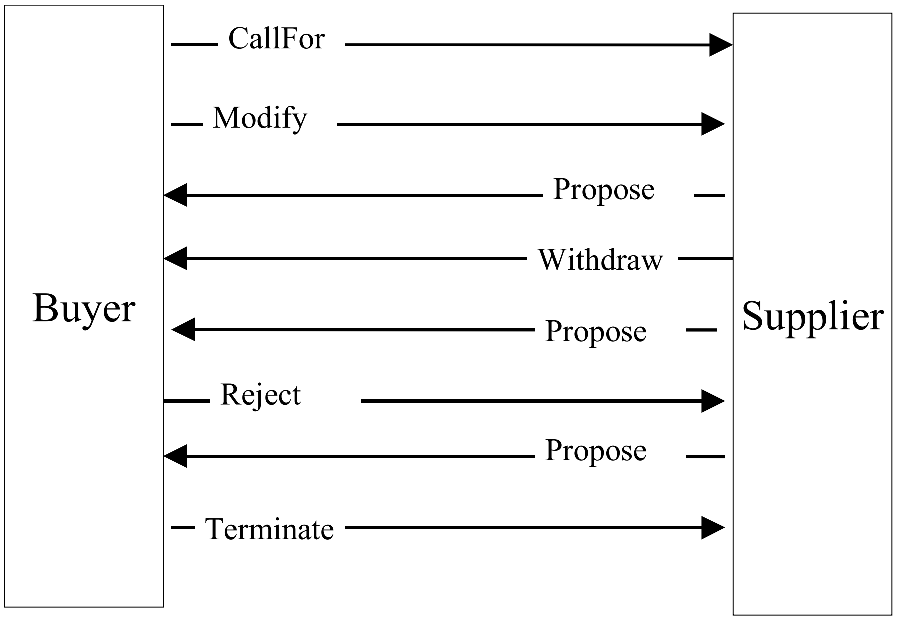

Figure 5. Withdraw/Modify scenario.

chronization problems between the two negotiation servers. If we call the negotiation server that sends a Modify/Withdraw as S and the receiver as R, it is possible that R may have responded to the previous proposal before it receives the Modify/Withdraw message (Synchronization Problem 1). To solve this problem, as shown in Figures 2 and 3, several state transitions have been added to the state transition diagrams so that the receiver (R) of Modify/Withdraw can reverse the state to process Modify/Withdraw, even if it has responded to the old proposal. Correspondingly, S needs to discard the obsolete message from R and wait for its new message. This can be achieved by letting the negotiation server transit to some states (e.g., S5, S11, S12 of Figures 2 and 3), in which all the incoming messages will be ignored (including the obsolete response message for the old proposal) after it receives the Modify/Withdraw request from a user.

In another case, the negotiation servers S and R may send a Modify/Withdraw message at the same time after S sent a proposal and R returned a counter-proposal; however, it is obvious that only S's Modify/Withdraw can take effect because R's counter-proposal has been invalidated by S's Modify/Withdraw. Therefore, the Modify/Withdraw of R's counter-proposal should be treated as invalid also (Synchronization Problem 2). To solve this problem, we design a mechanism to prevent both sides from sending Modify/Withdraw at the same time. In other words, Modify/Withdraw is similar to an exclusive token that only one negotiation server can use at one time. The mechanism works as follows. First, after a negotiation server receives a Propose, an Acknowledge message is returned to the sender if the receiver decides to process rather than to ignore the Propose message. (To simplify the diagrams given in Figures 2 and 3, we do not show the sending and receiving of Acknowledge messages). Second, a negotiation server can send a Modify or Withdraw message only after its proposal has been acknowledged. In this case, after the user has begun the Modify/Withdraw process by sending a ModifyReq/WithdrawReq to the negotiation server, all the incoming obsolete messages will not be acknowledged. Therefore, the negotiation partner can not send its Modify/Withdraw message before it processes the received Modify/Withdraw message.

8. System implementation

A negotiation server using BNOs has been implemented in Java programming language. IBM's XML Parser tools (ibmxml4j) are used to process BNODs. The server is installed in two separate computers, one represents a buyer and the other represents a supplier. The communications between these two servers are through Java RMI. In our implemented system, the negotiation process is triggered by a "Process RFQ" BOD received from an external system. The negotiation server on the buyer side generates a "CallFor Proposal" BNO based on the contents of the "Process RFQ" and sends it to the negotiation server on the supplier side. The CallFor Proposal is processed by the supplier's negotiation server, which generates a Propose Proposal based on the contents of the CallFor Proposal and supplier's registered information. The generated Propose Proposal is sent back to the buyer's negotiation server, which analyzes the received proposal and may generate a counter-proposal to the supplier. The negotiation process continues until either both sides accept the deal or one side unilaterally terminates the process. The system has been integrated with several commercial products (ERP and APT systems) and demonstrated in several consortium project meetings (www.ciimplex.org) as well as at the IEEE's International Conference on Data Engineering 2000 (Hammer et al. 2000).

The internal BNO processing is briefly described here. After the negotiation server receives a BNO, it is parsed and its contents are stored in a Constraint object. The received constraints are matched with the pre-registered constraints by a Constraint Satisfaction Processor (CSP). The CSP is able to identify which constraint is violated (e.g., the buyer asks for three days for delivery of the product, but the supplier can only deliver the product in five days). If a constraint violation is detected, an event that corresponds to the constraint is posted to trigger a negotiation strategic rule (if one exists), which may perform a concession by reducing the delivery days to four days (i.e., a concession by the supplier). Events and rules are managed by a server called the Event-Trigger-Rule Server (or ETR server). In this case, a counter-proposal may be generated by the supplier with the reduced delivery days and be sent to the buyer side. Since multiple values specified by a value range or an enumeration specification can be given for an attribute of a negotiation proposal, it is possible that multiple combinations of attribute values in a proposal all passed the constraint satisfaction test without any violation. This means that all the alternatives specified in the buyer's proposal do not violate the constraints of the supplier. In this case, these alternatives are evaluated by a cost-benefit evaluation component, which computes a global cost-preference score for each alternative based on the supplier's cost information and pre-registered preference scoring methods and aggregation functions. The supplier's negotiation server can then select the alternative that is most cost-beneficial to the supplier from those alternatives that the server represents, and either accept the best alternative or modify it to generate a counter-proposal. The CSP, the ETR server and the Cost-Benefit Evaluation server provide the knowledge and decision support needed for a negotiation server to conduct the bargaining type of negotiation on behalf of a client. For more detailed information about these servers and their functions, the readers are referred to (Su, et al., 2001).

9. Conclusion and discussion

The design and implementation of a set of business negotiation objects for supporting and implementing the bargaining type of business negotiations have been presented. These BNOs specify the operations and information contents needed for supporting the bargaining type of negotiations. They are a superset of the negotiation-related primitives defined in FIPA's ACL and COOL. We have also described the principles that were followed in our designs of BNOs and the system architecture before presenting the structures and contents of BNOs. The implementations of these objects (i.e., BNODs) are patterned after the standard XML BODs proposed by OAG. BNODs can be added to the existing set of common business object documents proposed by OAG for supporting the integration of heterogeneous application systems. They have been proposed to OAG for their inclusion into the set of XML BODs. The incorporation of several types of constraints in the BNO specifications is important in bargaining. The constraints associated with item attributes and transactional attributes extend the expressive power of call-for-proposals and proposals. The synchronization problems and their solutions associated with the use of the BNOs for Withdraw and Modify have been explained. These two primitives have not been proposed or implemented in any negotiation system.

There are several lessons learned in this R&D effort, and they motivate our future research efforts. First, it is not realistic to assume that two different business organizations (thus, their negotiation servers) will always use the same ontology and the same constraint language to specify their proposals and counter-proposals. In the Internet environment, it is likely that different organizations and their servers will have different ontologies and languages. There is a need to either define a common ontology/language for a specific business domain or provide a facility for mapping between ontologies/languages. An interlingua for the purpose of translation between different constraint specifications is also needed. The work reported in (Grosof and Labrou 1999) is an effort to solve this problem. Second, bargaining is a much more complicated process than the use of concession rules to generate counter-proposals as we have demonstrated in our past demonstrations. Rules, which govern other decision points in a negotiation process, such as the decisions to accept, reject, modify, or withdraw a proposal, and to terminate a negotiation transaction, need to be added by using the build-time tools provided by the ETR server to fully demonstrate the capabilities of the negotiation server. Third, negotiation strategies or tactics cannot be separated from business goals and policies of a business enterprise. The latter should determine the selection of the proper strategic rules used to drive the decisions and actions of a negotiation server. Also, the dynamic business conditions and past negotiation history must also be taken into consideration by the negotiation server in order to produce the best negotiation results for its clients. There is a need for a formal model of automated business negotiation which captures and relates business goals, policies, negotiation contexts, negotiation strategies, decisions and actions. The lessons learned from our past R&D efforts have helped us to identify the additional research problems discussed above. Our preliminary efforts in solving some of these problems are reported in (Liu et al. 2001; Li et al. 2001).

References

Adam, N. R., O. Dogramaci, A. Gangopadhyay, and Y. Yesha. (1999*). Electronic Commerce: Technical, Business, and Legal Issues*. Upper Saddle River, NJ: Prentice Hall PTR.

Austin, J. L. (1962). *How to Do Things with Words*. Oxford, England: Oxford University Press.

Barbuceanu, M., and M. S. Fox. (1995). "COOL: A Language for Describing Coordination in Multi-Agent Systems," *in Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, San Francisco, CA, pp. 17–24.

Breslin, J. W., and J. Z. Rubin. (1991). *Negotiation Theory and Practice*. The Program on Negotiation at Harvard Law School, Cambridge, MA.

Bui, T., F. Bodard, and P.-C. Ma. (1997). "ARBAS: A Formal Language to Support Argumentation in Network-based Organizations," *Journal of Management Information Systems* 14(3), 223–237.

Chang, M.-K., and C. Woo. (1994). "A Speech-Act-Based Negotiation Protocol: Design, Implementation, and Test Use," *ACM Transaction on Information Systems* 12(4), 360–382.

Collins, J., M. Tsvetovat, B. Mobasher, and M. Gini. (1998). "A Market Architecture for Multi-Agent Contracting," *in Second International Conference on Autonomous Agents*.

Conklin, J., and M. L. Begeman. (1988). "gIBIS: A Hypertext Tool for Exploratory Policy Discussion," *ACM Transactions on Office Information Systems* 6(4), 303–331.

Feldman, S. (1998). "Research Directions in Electronic Commerce," *in Third USENIX Workshop on Electronic Commerce*, Boston, MA.

Grosof, B. N., and Y. Labrou. (1999). "An Approach to using XML and a Rule-based Content Language with an Agent Communication Language," *in Proceedings of the IJCAI-99 Workshop on Agent Communication Languages (ACL-99)*, Stockholm, Sweden.

Hammer, J., C. Huang, Y. Huang, C. Pluempitiwiriyawej, M. Lee, H. Li, L. Wang, Y. Liu, and S. Y. W. Su. (2000). "The IDEAL Approach to Internet-Based Negotiation for E-Business," *in Proceedings of 16th International Conference on Data Engineering*, San Diego, CA.

Huang, C. (2000). "A Web-based Negotiation Server for Supporting Electronic Commerce," Ph.D. Dissertation, University of Florida, Gainesville, Florida.

Jackson, S., and S. Jacobs. (1980). "Structure of Conversational Argument: Pragmatic Bases for the Enthymeme," *Quarterly Journal of Speech* 66, 251–265.

Kalakota, R., and A. B. Whinston. (1996). *Frontiers of Electronic Commerce*. Reading, MA: Addison-Wesley, Inc.

Kersten, G. E., and S. J. Noronha. (1999). "WWW-based Negotiation Support: Design, Implementation, and Use," *Decision Support Systems* 25, 135–154.

Kunz, W., and H. Rittel. (1970). "*Issues as Elements of Information Systems*," Working Paper 131, Institute of Urban and Regional Development, University of California, Berkeley, CA.

Lee, J. G., J. Y. Kang, and E. S. Lee. (1997). "ICOMA: An Open Infrastructure for Agent-based Intelligent Electronic Commerce on the Internet," *in Proceedings of the International Conference on Parallel and Distributed Systems (ICPADS),* Seoul, Korea, pp. 648–655.

Li, H., S. Y. W. Su, H. Lam, and Y. Huang. (2001). "Automated E-business Negotiation: Model, Life Cycle and System Architecture," Technical Report, UF-CISE TR01-005. CISE Department, University of Florida, Gainesville, Florida. Available at: ftp://ftp.cise.ufl.edu/cis/tech-reports/tr01/tr01-005.pdf.

Lim, L., and I. Benbasat. (1993). "A Theoretical Perspective of Negotiation Support Systems," *Journal of Management Information Systems* 9(3), 27–44.

Liu Y., C. Pluempitiwiriyawej, Y. Shi, H. Lam, S. Y. W. Su, and H. Chan. (2001). "A Rule Warehouse System for Knowledge Sharing and Business Collaboration," Technical Report, UF-CISE TR01-006. CISE Department, University of Florida, Gainesville, Florida. Available at: ftp://ftp.cise.ufl.edu/cis/tech-reports/tr01/tr01-006.pdf.

Marriott, K., and P. J. Stuckey. (1998). *Programming with Constraints: An Introduction*. Cambridge, MA: MIT Press.

Müller, H. J. (1996). "Negotiation Principles," *Foundations of Distributed Artificial Intelligence*, *in* G. M. P. O'Hare, and N. R. Jennings (eds.), New York: John Wiley & Sons, Inc.

Myerson, R. B., and M. A. Satterthwaite. (1983). "Efficient Mechanisms for Bilateral Trading," *Journal of Economic Theory* 29, 265–281.

Pruitt, D. G. (1981). *Negotiation Behavior*. New York: Academic Press.

Raiffa, H. (1982). *The Art and Science of Negotiation*. Cambridge, MA: The Belknap Press of Harvard University Press.

Rangaswamy, A., and G. R. Shell. (1997). "Using Computers to Realize Joint Gains in Negotiations: Toward an 'Electronic Bargaining Table,' " *Management Science* 43(8), 1147–1163.

Reich, B., and I. Ben-Shaul. (1998). "A Componentized Architecture for Dynamic Electronic Markets," *SIGMOD Record* 27(4).

Rosenschein, J. S., and G. Zlotkin. (1994). *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. Cambridge, MA: MIT Press

Rubinstein, A. (1982). "Perfect Equilibrium in a Bargaining Model," *Econometrica* 50, 97–110.

Sandholm, T., and V. Lesser. (1995). "Issues in Automated Negotiation and Electronic Commerce: Extending the Contract Net Framework," *First International Conference on Multiagent Systems (ICMAS-95)*, San Francisco, CA, pp. 328–335.

Searle, J. E. (1975). "A Taxonomy of Illocutionary Acts," *in* K. Gunderson (ed.), *Language, Mind and Knowledge*, Minnesota Studies in the Philosophy of Science, Vol. 7. Minneapolis: University of Minnesota Press, pp. 344–369.

Searle, J. E. (1969). *Speech Acts: An Essay in the Philosophy of Language*. New York: Cambridge University Press.

Shepherd, A., N. Mayer, and A. Kuchinsky. (1990). "Strudel – An Extensible Electronic Conversation Toolkit," *CSCW'90 Proceedings*, New York: ACM Press, pp. 93–104.

Ståhl, I. (1972). *Bargaining Theory*. The Economics Research Institute, Stockholm, Sweden.

Su, S. Y. W., C. Huang, and J. Hammer. (2000). "A Replicable Web-based Negotiation Server for E-Commerce," *Proceedings of Thirty-third Hawaii International Conference on Systems Sciences (HICSS-33)*, Maui, Hawaii.

Su, S. Y. W., C. Huang, J. Hammer, Y. Huang, H. Li, L. Wang, Y. Liu, C. Pluempitiwiriyawej, M. Lee, and H. Lam. (2001). "An Internet-Based Negotiation Server for E-Commerce," *The VLDB Journal* 10(1), 72–90.

Tsang, E. (1993). *Foundation of Constraint Satisfaction*. New York: Academic Press.