The VLDB Journal 10: 72–90 (2001) / Digital Object Identifier (DOI) 10.1007/s007780100051

An Internet-based negotiation server for e-commerce

Stanley Y.W. Su, Chunbo Huang, Joachim Hammer, Yihua Huang, Haifei Li, Liu Wang, Youzhong Liu, Charnyote Pluempitiwiriyawej, Minsoo Lee, Herman Lam

Database Systems R& D Center, University of Florida, Gainesville, FL 32611–6125, USA; E-mail: su@cise.ufl.edu

Edited by F. Casati, M.-C. Shan, D. Georgakopoulos. Received: 30 October 2000 / Accepted: 12 January 2001
Published online: 2 August 2001 – © Springer-Verlag 2001

Abstract. This paper describes the design and implementation of a replicable, Internet-based negotiation server for conducting bargaining-type negotiations between enterprises involved in e-commerce and e-business. Enterprises can be buyers and sellers of products/services or participants of a complex supply chain engaged in purchasing, planning, and scheduling. Multiple copies of our server can be installed to complement the services of Web servers. Each enterprise can install or select a trusted negotiation server to represent his/her interests. Web-based GUI tools are used during the build-time registration process to specify the requirements, constraints, and rules that represent negotiation policies and strategies, preference scoring of different data conditions, and aggregation methods for deriving a global cost-benefit score for the item(s) under negotiation. The registration information is used by the negotiation servers to automatically conduct bargaining type negotiations on behalf of their clients. In this paper, we present the architecture of our implementation as well as a framework for automated negotiations, and describe a number of communication primitives which are used in the underlying negotiation protocol. A constraint satisfaction processor (CSP) is used to evaluate a negotiation proposal or counterproposal against the registered requirements and constraints of a client company. In case of a constraint violation, an event is posted to trigger the execution of negotiation strategic rules, which either automatically relax the violated constraint, ask for human intervention, invoke an application, or perform other remedial operations. An Event-Trigger-Rule (ETR) server is used to manage events, triggers, and rules. Negotiation strategic rules can be added or modified at run-time. A cost-benefit analysis component is used to perform quantitative analysis of alternatives. The use of negotiation servers to conduct automated negotiation has been demonstrated in the context of an integrated supply chain scenario.

Key words: E-commerce – Constraint evaluation – Cost-benefit analysis – Database – Negotiation protocol – Negotiation policy and strategy

Work funded by a NIST/ATP program grant.

1 Introduction

Internet and Web technologies have dramatically changed the way enterprises conduct business and how they compete. The Web provides customers and businesses with ubiquitous interactive access to text, sound, image, and video data. Millions of Internet users are potential consumers of online products and services. Thus, there are great incentives for companies of all sizes, ranging from multi-national corporations to small mom-and-pop operations, to automate interactions with their business partners, including suppliers and manufacturers [business-to-business (B2B) e-commerce] as well as their customers [business-to-customer (B2C) e-commerce]. In both paradigms, companies can receive purchase orders from individual consumers or business partners, deliver goods and/or services, and accept payments, all in an electronic fashion.

In ordinary, non-electronic business transactions, individual consumers and companies often want to negotiate the price, the delivery date, the quality of goods and services, as well as other purchase conditions; particularly if the order is large. There are three principal forms of negotiation: *bidding*, *auction*, and *bargaining*. Bidding is the simplest form of negotiation, in which a buyer specifies the product or service he[1] wants to acquire and asks for bids from potential suppliers. Based on the bids, the buyer selects the supplier(s) from whom to order the goods or services. The Contract Net Protocol (CNP) of Davis and Smith [DAV80] is a good example of bidding. Auction is another form of negotiation, in which a fixed auction protocol is followed, e.g., English auction, Dutch auction, and Vickrey auction [KUM98, BEA96]. Bargaining is the most complex form of negotiation, which may involve issuing proposals and counterproposals numerous times between negotiation parties until a mutual agreement or disagreement is reached. Several variations of bargaining also exist, e.g., bilateral bargaining, multilateral bargaining, single-issue bargaining (price), multi-issue bargaining (price, quantity, delivery time, etc.) [BEA97]. Different negotiation policies and strategies are often applied in different bargaining situations.

[1] The terms he, his, etc. are used throughout this paper to refer either to an individual (male or female) or a company.

Negotiation is an important business activity of an enterprise. Traditionally, negotiations (bidding, auction, and bargaining) are conducted by people involved in business transactions. However, in the context of e-commerce, in both B2B and B2C transactions, it is often desirable to carry out this negotiation process at least semi-automatically with human interventions only when necessary. To automate the negotiation process, two important tasks must be done: (1) formalize the negotiation process; and (2) incorporate necessary negotiation knowledge and intelligence into the computer system which carries out the negotiation. Formalization of the negotiation process enables the software to implement and automate the process, like any other type of automated business process, such as inventory management and purchase order processing. Incorporation of human and enterprise negotiation knowledge and intelligence enables a negotiation system to conduct automated negotiations effectively and intelligently on behalf of its clients.

Negotiation is a broad and complex problem that has attracted the attention of practitioners and researchers from many diverse areas, such as social sciences, communication, linguistic, politics, diplomacy, game theory, economics, etc. Recent interest in e-commerce has motivated a considerable amount of research work on automated negotiation systems. Commercial software packages and systems for negotiations have started to appear on the market. In spite of the numerous ongoing efforts focusing on various forms of negotiations, some of which are surveyed in Sect. 2, present research has not effectively solved the problems associated with the automation of the bargaining type of business negotiations. This motivated our research effort to develop methodologies and techniques for formalizing this type of negotiations and implementing a Web-based negotiation server to provide automated business negotiation services. In our work, the negotiation process is based on constraint satisfaction processing and relaxation of constraints through rules. A set of negotiation primitives and a formalized negotiation protocol are defined so that negotiation servers can effectively communicate and inter-operate with each other during a bargaining process. The knowledge and intelligence of the human negotiation experts is captured in the form of requirements, constraints, events, strategic rules, and preference scoring and aggregation methods. They are used by negotiation servers to conduct automated negotiations.

Based on the above techniques and methodologies, we have developed a replicable, Web-based, automated negotiation server for e-commerce applications. Multiple negotiation servers can be installed at different sites to provide negotiation services to registered companies. Enterprises that buy and sell products or services (henceforth referred to as clients) can install or select their own trusted negotiation servers to conduct negotiations on their behalf; the same way attorneys represent their clients in legal matters, for example. Analogous to Web servers, which provide many useful Web services, each negotiation server provides the following negotiation services:

- A *registration service* to allow clients to specify the requirements and constraints associated with the subject of their negotiation (e.g., goods, services, schedules, plans). In this paper, we shall use the buying and selling of products as our running negotiation scenario even though the negotiation server is capable of handling many other types of negotiations such as negotiations over delivery schedules, production plans, etc. A negotiation expert representing a client also registers a set of negotiation rules, which specify the negotiation strategies/tactics to be followed when constraints are violated during the negotiation phase. Additionally, he can specify the preference scoring and aggregation methods to be used for the cost-benefit analysis of the values under negotiation and the selection of the most cost-effective alternative.
- A *negotiation proposal processing service* to evaluate proposals, send messages (e.g., asking for additional clarifications or changes), and to generate counterproposals. In case of changes to the original proposal, explanations of the changes are also generated. All communication between negotiation servers follows a negotiation protocol.
- An *event and rule management service* to detect and manage events and to process rules that are relevant to the posted events. Rules may automatically relax the violated constraints, call for human interventions, invoke an application, or perform other remedial operations. The relaxed constraints as well as other external input (if available) form the counterproposals.
- A *cost-benefit analysis service* to allow a client to perform quantitative analysis on alternative negotiation data conditions to obtain their relative, quantitative cost-benefit measures. For this service, we have adopted the cost-benefit decision model presented in [SU87] and implemented a subset of its preference scoring and aggregation methods.

We have already reported on an earlier version of the negotiation server, which was based on the 1998 version of the implementation [HAM00; SU00]. Here we present the latest version of the server. Our main contributions are: (1) the formalization of the bargaining type of automated negotiations (i.e., negotiation primitives and protocol); and (2) the integration of a number of technologies such as object-oriented requirement and constraint specifications, constraint satisfaction processing, event-trigger-rule processing, and cost-benefit evaluation and selection in the implementation of a Web-based negotiation server. This is a system-oriented paper. As such, all aspects and functions of the implemented system are covered. However, space limitations do not allow us to present technical details of all the aspects and functions of the system. We shall focus only on negotiation primitives, the protocol, and the constraint satisfaction processing and its relationship with the event-trigger-rule server. Interested readers are referred to [LAM98, RAG99, HUA00, LEE00] for more technical details.

The remainder of the paper is organized as follows. Section 2 surveys the related work. In Sect. 3, we present the architecture for our Internet-based negotiation server as well as a framework for conducting automated negotiation. In Sect. 4, we describe the constraint, event, rule and trigger specifications of an object-oriented content specification language by examples taken from a recent negotiation demonstration in the context of a supply chain management application. Section 5 provides the details of the negotiation process, the algorithm used in constraint satisfaction processing, and the underlying negotiation protocol. Section 6 outlines our cost-benefit decision model. Section 7 describes the implementation of our negotiation server and Sect. 8 concludes the paper with a summary and a discussion on the issues which we have encountered during the course of this research.

2 Survey of related work

We group the existing work in the following six categories.

2.1 Social sciences

Pruitt [PRU81] has studied negotiations from a social-psychological point of view. His book deals with human psychology that is involved in face-to-face negotiations. The work reported in Raiffa's book [RAI82] divides negotiations into several categories based on the number of parties and issues involved: two parties/one issue, two parties/many issues or many parties/many issues. According to Raiffa, different categories of negotiations raise different problems. For example, coalition formation is not a problem when only two parties are involved. However, it is one of the most important topics in multiple-party negotiation. Although the book is written for face-to-face negotiation among people, basic negotiation principles are equally applicable to the development of an automated negotiation server. They are especially relevant to the specification of strategic rules used by the server. Several other books [LAX86, KAR93, SHE99] offer practical negotiation advice for negotiators. The Program On Negotiation (PON [PRO00]) at the Harvard Law School and the Stanford Center on Conflict and Negotiation (SCCN [STA00]) at Stanford University focus on social and legal aspects of negotiation. Although their research is social science in nature, it touches upon the topic of using computers to aid and even automate the negotiation process.

2.2 Game theory

Game theory [BIN99, JON80] is the mathematical study of conflicts. Since negotiation is one type of conflict, game theory has been used to analyze negotiation processes for a long time. It focuses on the predication of whether or not an agreement will be reached and, if so, what the specific nature of that agreement is. Game theory usually assumes that the participants (players) are rational and have complete information about the other players and their expected behavior. Research in game theory also focuses on mechanism designs: the definitions of protocols that limit the possible tactics or strategies used by players and the mechanisms to achieve the so-called Pareto outcome for all negotiation participants using the values of a utility function [ZLO96].

2.3 Negotiation support systems

Negotiation Support Systems (NSS) [LIM93, YAN00] and Group Decision Support Systems (GDSS) [KAR97] extend Decision Support Systems (DSS) to the area of negotiation. The challenges of negotiation and the shortcomings of human negotiators have prompted researchers to pursue computer-supported negotiations. Jelassi and Foroughi [JEL89] have called for tools which address behavioral characteristics and cognitive perspectives of negotiators. Woo [WOO90] uses speech act theory to formalize the negotiation process so that machine transmission of messages is possible. Matwin, Szapiro, and Haigh [MAT91] introduce a concession model of negotiation, which hard-wires a general strategy of concession-making into a multi-issue negotiation system.

A very different NSS developed by Rangaswamy and Shell [RAN97] employs a computer-based method to elicit a conjoint representation of preferences. Once the parties have a better understanding of their preferences, they make proposals electronically. In controlled experiments, supported users reached better agreements. An additional feature of the system is that it observes the offers made by each party and, by knowing the preferences of both, can suggest a Pareto equilibrium for improved outcomes. Negoplan is a decision support software system developed by the International Institute for Applied System Analysis (IIASA) in Austria [KER99]. It is implemented in Prolog and simulates decision processes by allowing a systematic and analytical solution of sequential decision problems, of which negotiation is an example.

2.4 Agent technologies

Agent technologies have been widely used by the computer community to develop systems that can replace some of the intelligent activities of human beings. Negotiation is one of such activity. Distributed Artificial Intelligence (DAI) [OHR96, MUL96] and Multi-Agent Systems (MAS) [ZLO96], as two main branches of agent technology research, try to automate the negotiation process between agents. Sandholm [SAN96] has studied the coalition formation problem (a complex form of negotiation) in the context of distributed artificial intelligence and multi-agent systems. Kasbah [CHA96, CHA97, KAS99] is a Web-based, multi-agent, classified ad system where users create buying and selling agents to exchange goods. These agents automate much of the merchant brokering and negotiation for both buyers and sellers. A user wanting to buy or sell an item creates an agent, inputs his strategic directions, and sends it to a centralized agent marketplace. Kasbah's agents proactively seek out potential buyers or sellers and negotiate with them on behalf of their owners. Negotiation between buying and selling agents in Kasbah is bilateral and straightforward. After matching the corresponding buying and selling agents, the only valid action for buying agents is to offer a bid to a seller. Selling agents respond with either a binding "yes" or "no". Given this protocol, Kasbah provides buyers with one of three *negotiation strategies:* "anxious", "cool-headed", and "frugal" – which correspond to a linear, quadratic, or exponential function, respectively, for increasing the bid amount for a product over time. The simplicity of these heuristics makes it intuitive for users to understand how their agents behave in the marketplace.

2.5 Machine learning

Rather than attempting to exhaustively translate negotiation strategies from humans to software agents, the field of machine learning attempts to let software agents learn how to negotiate among themselves. Zeng and Sycara present Bazaar [ZEN98], an experimental system for updating negotiation offers between two intelligent agents during bilateral negotiations. The paper contains a formal analysis of the negotiation state space, which is capable of tracking a rich set of issues and trade-offs that are necessary for multi-issue negotiations. It explicitly models negotiation as a sequential decision making task, and uses Bayesian probability as the underlying learning mechanism.

Another learning approach is to use genetic algorithms and genetic programming [DWO96, OLI97]. In the context of negotiation, it works as follows. Each of the software agents begins with a population of various, randomly generated (and not necessarily very good) negotiation strategies. It then employs its strategies against the strategies of the other agents in a round of bargaining, which takes place under specific predetermined rules and payoffs. At the end of a "generation" the agent evaluates the performance of each strategy in its current population, and crosses over strategies from the current "parent" population to create a "child" generation of bargaining strategies. The more successful strategies are chosen to be parents with a higher probability; also, mutations may be randomly introduced. The size of the initial population, the number of generations, the crossover rate, and the mutation rate are parameters of the algorithm.

2.6 Commercial software packages and systems

Priceline [PRI00] is an example of a reverse auction system. In a normal auction, the seller sets an initial (usually low) price, and lets the buyers compete for the product by successively increasing the bids. In a reverse auction, the buyer sets the price and lets the sellers match it. Traditional Request-For-Quotes (RFQs) or Request-For-Proposals (RFPs) are processed manually. Perfect [PER00] has developed an online RFQ processing engine. It provides buyers with tools that allow them to concentrate on defining their needs precisely and evaluating only the most relevant offers. It dramatically reduces procurement costs by automating the manual process of searching for and requesting quotes and by eliminating non-competitive offers.

HaggleWare [HAG00] provides pricing decisions that match buyers and sellers who are engaged in real-time, online negotiations. Through HaggleWare, customers can make price offers. The system creates unique profiles for all buyers and their price bids are submitted into the core engine. Product information such as quantity, pricing, and duration are transmitted from the sellers to the core engine at the same time. The software then analyzes buyers' offers against the sellers' product information and gives instant feedback to buyers. The buyers can react to counter-offers, seek quantity discounts, and even threaten to "walk away" in case the negotiation stalls. HaggleWare is a semi-automated negotiation system, which focuses on price negotiations. Only the seller side is automated. MakeUsAnOffer [MAK00] is a Website that uses the HaggleWare technology to build a "Real-Time Online Haggling" market. HaggleWare is installed on the seller's side and the buyer can negotiate with fictitious sales representatives.

Win Square [WIN00] is another software package to help the negotiator find the best negotiation strategy. The package can: (1) recommend strategies and tactics for the user; (2) predict what strategies and tactics the other party will use; and (3) recommend defenses to the other party's tactics.

All of the above systems support only bi-lateral negotiations. One Accord Technologies [ONE00] has a range of products to support both bi-lateral as well as multi-lateral negotiations. As a stand-alone system, One Accord Negotiator can help users better understand the issues and their own preferences. The user can also simulate other parties and develop strategies for negotiation. Using the One Accord Network, Negotiator puts the user in secure real-time communication with other Negotiators and generates optimal solutions based on the preferences of other negotiation parties located anywhere on the Internet. The Pro version is designed to facilitate negotiations in either the stand-alone mode or through the One Accord Network.

In summary, the existing research and commercial software have already provided great insights into negotiation policy and strategy, decision-making, negotiation protocols, and automation of the negotiation procedure. However, they have not dealt with the architecture, methodology, formalism, and implementation of an automated negotiation system for supporting **bargaining type business negotiations** as is addressed here.

3 Negotiation server architecture and negotiation framework

We now present our system architecture and framework for negotiation. Figure 1 shows the relationship among the negotiation servers, Web servers, client computers, and humans in our proposed negotiation framework. In addition, some of the client companies may have application systems (e.g., ERP systems) maintaining their inventories of goods or information about the services they provide. To simplify our presentation of the negotiation process, we shall use the symbols EA, UA, and NSA to denote the expert, user, and negotiation server of negotiation party A. The corresponding symbols EB, UB, and NSB are used to denote those of negotiation party B.

In our framework, sellers can publish information about the goods and services they provide on their home pages, which can be browsed by potential buyers using Web browsers and search engines. Through searching and browsing, or by using available trader services, a buyer can identify potential sellers with whom to conduct negotiations. In order for buyers and sellers to make use of the automated negotiation service, each party can either install and register with a negotiation server by providing the information needed by the server to conduct automated negotiations on its behalf (see Sect. 3 below), or establish a trust agreement with an installed negotiation server and provide it with the registration information. The registration is done by humans (i.e., EA) who are responsible for forming the negotiation policies and strategies of the party they represent and for specifying product/service requirements as well as constraints.

A user (i.e., UA), who uses and monitors the progress of a negotiation server, initiates the negotiation process by submitting an initial product or service request to his negotiation server. The negotiation server (NSA) initiates a negotiation process by creating and sending a computerized, XML-wrapped *call-for-proposal* (CFP) to NSB, which represents

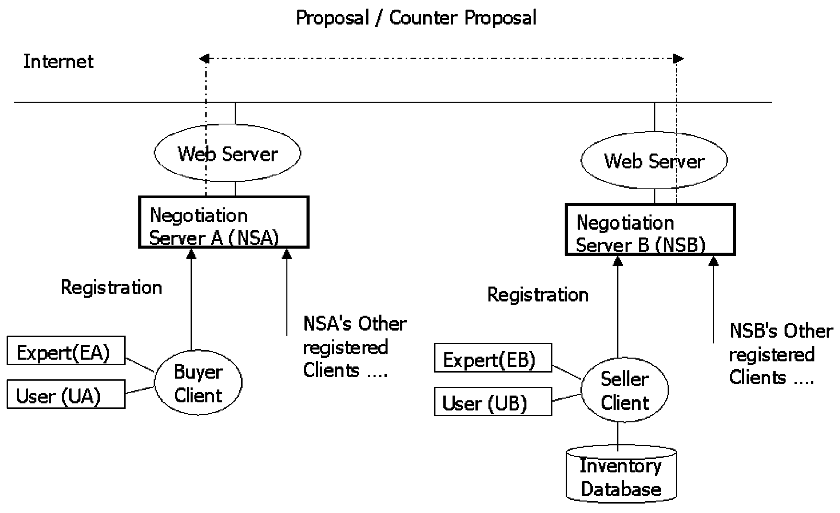

Proposal / Counter Proposal
Internet
Web Server
Web Server
Negotiation Server A (NSA)
Negotiation Server B (NSB)
Registration
Registration
NSA's Other registered Clients ....
NSB's Other registered Clients ....
Expert(EA)
User (UA)
Buyer Client
Expert(EB)
User (UB)
Seller Client
Inventory Database

Fig. 1. An overview of Internet-based negotiation

the seller. In case the seller maintains an inventory of goods or recorded information on services (e.g., a database system which is available to the negotiation system), NSB, can use the data and constraints of the proposal to query the database to verify the availability of the goods or services in question. NSB also checks the information and constraints specified in the CFP against the registered information of EB to determine if there are conflicts in requirements and constraints. The conflicting data conditions and constraints can be modified either by registered rules of NSB or through human intervention (UB) to produce a proposal to NSA. The proposal will then be evaluated by NSA against the registered information of its client.

In the event that a conflict or constraint violation is found, relevant strategic rules, which have been provided by EA, are applied to either: (1) reject the proposal and suggest possible changes to NSB; (2) consult with UA for decision-making; or (3) generate a counterproposal to be returned to NSB. In the event that the proposal contains a number of alternative data conditions (see Sect. 5), a cost-benefit analysis component is invoked to perform quantitative analysis on each alternative and to provide a cost-benefit rating. The highest ranked alternative will be sent back to NSB together with EA's constraints as the counterproposal, which starts another round of negotiation. The counterproposal is evaluated in the same way by NSB against its client's registered requirements and constraints. This process will continue until an agreement is reached or either side terminates the negotiation process. The above description represents a very simple bargaining process. More complicated scenarios involving the exchange of various negotiation primitives are shown in Sect. 5.

In summary, our system architecture has the following characteristics.

- The architecture is *Web-based*. The negotiation server can be installed in cooperation with existing Web servers and provides negotiation services to their client companies on the Internet. Clients can install or select trusted negotiation servers to conduct negotiations on their behalf. Each negotiation server will interact with other designated negotiation server(s) to carry out the assigned negotiation tasks. The negotiation process is conducted automatically unless human intervention is necessary.
- The architecture is *open*. The architecture takes full advantage of the Web as the largest single information resource. Negotiation parties can locate each other using search engines or other, third-party information brokers or a marketplace [LEE97]. There is no need for a centralized marketplace during or after the negotiation. Please note that, although replicated negotiation servers handle separate registration processes and manage the registered information, the architecture does not exclude the use of an external catalog management system that contains information about products or services.
- The architecture is *general and flexible*. It can support multiple negotiation types, e.g., auction, bidding, and bargaining. For instance, in bargaining, both negotiation servers will implement a bargaining protocol and negotiate with each other; in an auction, one server will be the auctioneer and other servers can represent the bidding clients. In the latter case, an auction protocol can be used instead of the bargaining protocol.
- The architecture is *scalable* in the same sense as the Web is scalable. The negotiation server can be replicated like a Web server and can be attached to a large number of available Web servers. Large business organizations can install their own negotiation servers and small business organizations or individuals can establish a trust agreement with installed server sites and gain access to the automated negotiation services that way. Scalability is also achieved by each negotiation server's ability to process multiple negotiation transactions concurrently and by its registration service for registering new clients.

4 Registration and content specification language

Before automated negotiation can take place, a client must first inform the corresponding negotiation server, and indirectly the other parties involved, of the goods and services he provides (in the case of a supplier) or wishes to acquire (in the case of a consumer) as well as any special requirements that may influence the negotiation (e.g., delivery constraints or special discounts for large orders). This information is captured in the form of *registration* information during the initial registration

```
ENTITY Computer_System {
ATTRIBUTE-CONSTRAINT
   model      String    ENUMERATION{PII350, PII400}     priority [3]
   memory     Integer   ENUMERATION {32m,64m, 96m}      priority [4]
   monitor    Integer   ENUMERATION {17, 19}            priority [5]
   hard_drive       Integer    ENUMERATION {4g, 6g, 8g} priority [6]
   unit_price       Float      DERIVED                  priority [2]
   deliver_day      Integer    RANGE[14..21]            priority [1]
   quantity         Integer    RANGE [250..550]         NotNegotiable
INTER-ATTRIBUTE-CONSTRAINT
  quantity_deliver_day_1  quantity >= 400 implies  deliver_day >= 16  priority [1]
  model_memory_1          model = 'PII400' impliesmemory >=64m       priority [2]
}
```

process. For suppliers, part of the registration procedure is done in the form of advertisements, which are published as Web pages. The contents of these advertisements can be indexed and catalogued, for example, by application domains, and are searchable like regular Web pages. We assume that negotiation servers either maintain their own index of relevant advertisements or use a third-party search engine (e.g., AltaVista) or yellow page service (e.g., Yahoo) to help locate the advertisements (i.e., the pull model of information access). Alternatively, one can also envision a scenario in which suppliers actively inform the relevant negotiation servers of their goods and services by sending the advertisements directly to the servers (i.e., the push model). The approach described in this paper is independent of the particular paradigm (i.e., either push or pull) used by suppliers. In our current implementation, the information push model is used. For buyer companies, the registration is done by specifying a set of requirements and constraints for describing the desired goods or services and rules for expressing negotiation strategies to their own negotiation servers. Negotiation rules can be added and modified at run-time to deal with the dynamic nature of negotiations. Another important part of the registration information is the data to be used by the cost-benefit decision model, which is discussed in Sect. 6.

In order for a negotiation server to communicate with the negotiation experts of its client and acquire registration information from them, the server must provide them with a common language or GUI tool for the registration purpose. Such a language is also useful for formulating negotiation proposals, counterproposals, and other negotiation messages. In our work, we have developed a content specification language and a set of Web-based GUI tools for expressing the registration information, i.e., attributes, data values, constraints, and strategic rules. The syntax of the language and the forms and structures used in the GUI tools are simple to design. However, the semantics of the words and word relationships used in the language and GUI tools present a difficult problem because different companies and users may use different words to mean the same things and same words to mean different things. This is the so-called *semantic interoperability* problem faced by all heterogeneous information systems. There are two commonly recognized solutions to this problem. One is to establish a common ontology [FOX94] for each domain so that all parties in the same domain speak and understand the same ontology. This approach is taken by several standards groups such as Commerce One, Open Applications Group, RosettaNet, etc. The second approach is to use mapping or mediation to resolve the different uses of words. In this work, we assume that a common ontology for a particular product or service domain has been established and is used by the parties involved in the negotiation.

Negotiation items are modeled as active objects in an Active Object Model (AOM) [LEE00]. In addition to the traditional object specification using attributes and methods, an active object may have events, triggers, action-oriented rules and constraints associated with it. We shall use examples to describe the constraint, event, and rule and trigger specifications of the content specification language, and defer a detailed description to [HUA00].

4.1 Data and constraint specification

Registration information is entered through a graphical forms-based interface, which is provided to the user through servlets. The registered information is then stored as objects in a persistent repository, which is part of the negotiation server. For example, a product or service is represented as an object, which has attributes describing its properties and constraints that must hold for individual attributes (attribute constraints) or a number of attributes (inter-attribute constraints). The following example depicts a sample advertisement of a computer system offered by a supplier. Consumer registration information can be represented analogously.

In the example on top of this page, the supplier advertises a computer system, which is specified as an `ENTITY object class` named `Computer_System`. The advertised entity class is available in many different configurations as described by the attributes. Each attribute value is of a particular (atomic) type (e.g., String, Integer). An attribute constraint is specified by *enumerating* a set of possible values (note that a single value is a special case of an enumeration) or by a value *range*. In addition, attributes whose values depend on other values and cannot be determined until the negotiation is under way are marked as `DERIVED`. The supplier has its own formulas for computing the values of the derived attributes. All attribute values are negotiable unless marked by the special keyword NotNegotiable. In addition to attribute constraints, constraints associated with attributes of the same or different object classes are called inter-attribute constraints. The syntax for inter-attribute constraints is:

`constraint-name antecedence` or
`constraint-name antecedence` **implies** `consequence`

In the example above, the inter-attribute constraint called `model_memory_1` specifies that if a customer opts for a computer model 'PII400', the corresponding memory size must be at least 64 MB. The priority numbers are used for determining the order in which constraints are validated starting from the smallest. However, constraints marked as NotNegotiable will be checked first.

Associated with each attribute or inter-attribute constraint is an implicitly defined event. The name of the event is generated by extending the names of an entity object class, and the attributes or inter-attribute constraints. The event triggers rules when the violation of the attribute or inter-attribute constraint is detected by the Constraint Satisfaction Processing component. Events can also be defined explicitly and can be posted before and/or after method calls. Events can have parameters whose values are passed to rules for verification of data conditions.

4.2 Rules for specifying negotiation strategies

An important part of the registration procedure is the specification of negotiation strategies or tactics to be used by the negotiation server on behalf of its client. In our negotiation system, we adopt a declarative approach. Each strategy is expressed in terms of an event-trigger-rule (ETR) specification, which states the condition to be checked and the actions to be taken by the negotiation server. Rules are activated upon the occurrence of specific events.

A strategic rule has three parts: (1) the condition under which the subsequent action is to be taken (e.g., the proposed sales price is below manufacturing cost); (2) the particular action(s) to be taken when the condition is met (e.g., the modification of the current constraint); (3) an optional alternative action(s) to be taken when the condition is not met (e.g., a request for human intervention). The condition part can be a complex Boolean expression and all three parts can contain method calls to local or remote objects. The relationships between events and CAA rules are specified by triggers. A trigger specifies an *event structure* and a *rule structure*. An event structure has two parts: TRIGGEREVENT and EVENTHISTORY. The TRIGGEREVENT part specifies a number of events called triggering events, each of which when posted, triggers the evaluation of the event history specified in the EVENTHISTORY part. If the event history is evaluated to true, the structure of rules specified in the RULESTRUC clause is processed. Otherwise, the structure of rules will not be processed. EVENTHISTORY is used to specify the relationship between some events that have already occurred or posted. An event history expression given in this clause is called a "composite event" in the active database literature. It can contain a number of event names with logical or sequence operators. For example, the expression "*OrderShipped* and *OrderException*" states that both events must have been posted if the rules specified in RULESTRUC are to be fired. The expression (E3 > E2 > E11) specifies that these three events must have occurred in the given sequence. The data structure and algorithm used in the maintenance and processing of event history and the implementation of the ETR server are out of the scope of this paper. Interested readers should refer to [RAG99, LEE00] for details.

The TRIGGEREVENT part is purposely kept simple. Only the occurrence of any one of the triggering events triggers the evaluation of a more complex event history expression. The separation of TRIGGEREVENT and EVENTHISTORY allows more explicit specification of what events would trigger the evaluation of event history and rules. This is different from the event specification of other existing ECA rule systems in which, when a composite event is specified, all of the events mentioned in the composite event implicitly become the triggering events. In some applications, one may want to specify that only the posting of E2 should trigger the evaluation of "E1 and E2 and not E3". The RULESTRUC clause of a trigger specification allows a structure of CAA rules to be triggered when the event specification is satisfied. It specifies the rule execution order and maps the parameters of the trigger event to the individual rules. There are two operators for specifying the execution order: the operator '>' is used to specify a sequential order of rule execution, and the operator ',' is used to specify a parallel execution. For example, the expression (R1 > R2 > R3 > R4) specifies that rules R1, R2, R3, and R4 are to be executed sequentially and the expression (R1, R2, R3, R4) specifies that these rules are to be executed in parallel. The specification of a network structure of rules with fan-in and fan-out branches is also allowed. A CAA rule specifies a small granule of logic and control, whereas a rule structure specifies a larger granule of logic and control.

The separation of events, rules, and triggers provides more flexibility in linking trigger events and event histories with rules or rule structures than the traditional Event-Condition-Action (or ECA) rules. For example, a rule name may appear in several rule structures, which are triggered by the occurrences of different events. In a traditional ECA system, such a rule has to be specified repeatedly using multiple ECA rules. Furthermore, the ETR specification makes the semantic distinction between trigger events, the verification of composite events or event history, and the rule structure more explicit than the traditional ECA rules. To our knowledge, all current ECA systems treat the events of a composite event as trigger events, i.e., the occurrence of any one of these events will trigger the evaluation of the composite event. However, in some situations, only some of these events should trigger the evaluation of the composite event and associated rules. For full descriptions of the AOM and the ETR server, please refer to [LAM98, LEE00].

The following two strategic rules on the supplier side outline a very simple specification of a negotiation strategy using the ETR format. We refer to the entity class `Computer_System` in the sample advertisement given above as the context. In the example, events are posted by the Constraint Satisfaction Processing component of the negotiation server upon detecting a conflict or violation of an attribute constraint or an inter-attribute constraint. The event names are generated by extending the corresponding constraint and attribute names. Since the sample negotiation rules are very simple (no composite events) we only show trigger events and rules that constitute trigger specifications instead of using the full syntax of our content specification language.

Strategic Rule 1: If `deliver_day` is out of range, check the lower bound of the constraint for attribute deliver_day. If the lower bound is still greater than 10 (i.e., the bottom line), shorten the delivery time by two days. Otherwise, the constraint is considered unResolvable and human intervention is required.

```
TriggerEvent         SupplierComputer_Systemdelivery_day
SR1:   Condition:    getLowBound("delivery_day") > 10
       Action:       downLowBound("delivery_day", 2);
       Alternative:  unResolvable("delivery_day can not
                                be satisfied")
```

Strategic Rule 2: If the *quantity_deliver_day_1* constraint is violated, check if this constraint has already been relaxed. If not, relax the constraint to: "quantity≥500 **implies** delivery_period≥15"; change the relaxation flag to true in order to prevent further relaxation. Otherwise, the constraint is considered unResolvable and human intervention is required.

```
TriggerEvent       SupplierComputer_Systemquantity
                          _deliver_day_1
SR2: Condition:    getIACStatus("quantity_deliver_day_1")
                          = "NOCHANGE"
     Action:       setIAC("quantity_deliver_day_1",
                          "quantity >= 500",
                          "delivery_period>=15");
                   setIACStatus("quantity_deliver_day_1",
                          "RELAXED");
     Alternative: unResolvable("delivery_period can not
                                be satisfied");
```

In our implementation, strategic rules for negotiations can be entered through a GUI at registration time or added and modified at run-time. Rules are translated into Java rule classes when they are defined. The class reloading feature of Java is used to dynamically replace the old rule classes by new rule classes if rules have been modified at run-time.

5 Negotiation process

We now describe three important aspects of the negotiation process: (1) the negotiation primitives, which define messages passed between negotiation parties, (2) the negotiation proposal, which expresses negotiation conditions and constraints, and (3) the procedure and algorithm for processing the proposal and for generating a counterproposal. The last step includes constraint checking, triggering of strategic rules, inventory verification, and cost-benefit analysis.

5.1 Negotiation primitives

Table 1 shows a list of negotiation primitives used by our system. It overlaps with the list of negotiation primitives given in [MUL96], which includes additional primitives useful for agent communication but not germane to this work. It is a superset of the set of negotiation primitives proposed by FIPA [FIP97], which contains only the primitives CFP, Propose, Reject, and Accept.

5.2 Negotiation proposal

A user initiates a negotiation process by sending a call-for-proposal (CFP) via its negotiation server to inform the other side about his intentions, requirements, and constraints with respect to the item that is to be negotiated. The CFP will be evaluated by the negotiation server of the supplier side against the supplier's own registered general requirements and constraints. It forms the basis for the proposal that the supplier returns. It is worth noting that some of data on the CFP may be modified before being included in the proposal depending on the constraints of the supplier. The returned proposal will be evaluated against the buyer's registered requirements and constraints. If the buyer does not agree with the conditions and constraints specified in the proposal, the server of the buyer may modify its contents and return it as a counterproposal. This exchange of counterproposals between the two servers may continue until a final agreement is reached, i.e., one side accepts the counterproposal without any further modifications, or until one side terminates the negotiation process.

Table 1. Negotiation primitives

CFP	Initiate a negotiation process by calling for proposals
Propose	Issue a proposal or a counterproposal
Accept	Accept the terms and conditions 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

Existing negotiation systems allow only constant values to be specified in a proposal. For example, in a proposal describing a task allocation scenario, the negotiated task must have fixed values for resource requirements, time deadline, cost, etc. In another example, when buying a TV, the proposal must provide values for the size, brand name, price, etc. of the desired TV. The disadvantages of this restriction are obvious. First, a client may not have the domain knowledge to provide a value for each attribute. For example, in our sample negotiation over a computer purchase, the buyer has no idea about the particular warranty contract that is available for the desired product. In this case, it should be possible to leave the value of this attribute unspecified. Second, a client may not have enough knowledge to compute values that depend on other (unknown) values. For instance, determining the proper CPU/memory configuration is a complicated decision task for most customers. Instead, clients should be allowed to specify ranges of values or enumerate alternatives for these attributes. In our approach, a call-for-proposal or a proposal/counterproposal contains the data and constraints specifying the goods a buyer or seller wants to acquire or provide, respectively. We use the same content specification language to specify proposals and counterproposals as in the registration phase for specifying general requirements and constraints. Proposals are translated into XML business object documents as proposed by the Open Application Group [OPE95] before transmission over the network.

```
Entity Proposal{
ATTRIBUTE-CONSTRAINT:
          model        String   ENUMERATION{''PentiumII 350"}
          monitor      Integer  ENUMERATION{15, 17, 19}
          memory       Integer  RANGE[16m..64m]
          hard_drive   Integer  RANGE [1g.. 12g]
          unit_price   Float    ENUMERATION{1700.00}
          deliver_day  Integer  RANGE[3..10]
          quantity     Integer  RANGE[300..400]
INTER-ATTRIBUTE-CONSTRAINT
          Constraint1     memory < 64m implies hard_dirve < 4g
          Constraint2     monitor = 15 implies unit_price < 1500
}
```

The above example shows a sample counterproposal (without XML tags) issued by the buyer's negotiation server during the negotiation of the computer purchase (see top of this page).

Abstractly speaking, a proposal represents a hierarchy consisting of attributes and constraints. Each composite attribute is represented by a substructure of attributes and their constraints. Since range and enumeration are used to specify the constraints of some or all of the attributes, a proposal specifies many combinations of data values that represent the acceptable alternatives of the same product. A proposal is an instance of a proposal object class. The data, attribute, and inter-attribute constraints of the proposal instance are processed by the Constraint Satisfaction Processing component of the negotiation server against the registered data and constraints of the recipient. Comparison operators for Range and Enumeration are introduced for processing the data. During processing, conflicts or violations of constraints will result in the posting of the corresponding events. Note that the proposal and registration information differ mainly in the fact that proposals do not include strategic rules. In addition, the constraints in a proposal must be consistent with the registered constraints of the client issuing the proposal.

5.3 Negotiation procedure

After receiving a proposal, a negotiation server has the following choices: accept it, respond with a counterproposal, reject it with an explanation which is optional, or terminate the negotiation. In order to arrive at a decision, the negotiation server follows the general proposal processing procedure as described in Fig. 2. We first describe the different processes and their interactions as shown in the figure and then focus on the theory behind our approach to constraint processing which is one of the focal points of this research.

The Constraint Satisfaction Processing (CSP) component in Fig. 2 is used to evaluate the incoming proposal against the client's registered information. As we can see, attribute constraints are evaluated first before evaluating inter-attribute constraints. The evaluation of attributes and attribute constraints follows the priority order of the registered information as illustrated in Sect. 3.1. If a conflict is found during this process, the event that corresponds to the attribute will be posted to the ETR server to trigger the evaluation of an event history and the associated rule or rule structure. The rule or rule structure may automatically relax the violated constraint (e.g., to reduce the delivery date by 2 days to meet or to come closer to the buyer's desired delivery date) or call for human intervention to relax the constraint. The relaxed value can be used directly in the counterproposal to be sent back to the negotiation server of the other party. In this case, the client's strategy is to give in little by little (i.e., one violation at a time) in an attempt to meet the buyer somewhere in the middle. Alternatively, the evaluation of attributes and attribute constraints can continue until a pre-specified number of conflicts (this depends on the negotiation strategy) are found. The conflicting constraints are relaxed to form the counterproposal. This alternative strategy would speed up the negotiation process but may not necessarily result in an optimal result for the client who may relax more constraints than necessary.

If no conflict is found during constraint evaluation (meaning, for each attribute, the range or enumeration of values of the received proposal overlaps with those of the registered data), a set of negotiation item instances is created by taking the combinations of the overlapping attribute values or value intervals. Each of these instances represents a negotiation item description that satisfies the attribute constraints of both negotiation parties. The set of instances is then evaluated against the inter-attribute constraints that come with the proposal, in order to filter out those that do not match. The remaining set is then evaluated against the registered inter-attribute constraints of the client by following their priority order. Again, if a violation of any inter-attribute constraint is found (i.e., no product instance satisfies the constraint), a corresponding event is posted and a strategic rule (if available) is triggered to either relax the constraint automatically or call for human intervention. If no rule is available, the constraint violation may cause the rejection of the proposal or the termination of the negotiation process.

After the inter-attribute constraint evaluation, if no violation is found (meaning, there exists at least one negotiation item instance that satisfies all the attribute and inter-attribute constraints), the proposal is acceptable to the client. In that case, an `Accept` message is sent to the other party. In case multiple negotiation item instances satisfy all the constraints of the client, Cost Benefit Analysis Module (see Sect. 6) is invoked to quantitatively evaluate and rank the alternatives. The negotiation item description that is most beneficial to the client is selected.

We shall explain the constraint evaluation procedure further by using our computer purchase example. The buyer's proposal given in Sect. 4.2 is processed by the negotiation server NSB against the supplier's registration information given in Sect. 4.1. First, attribute-constraints are checked. Ac-

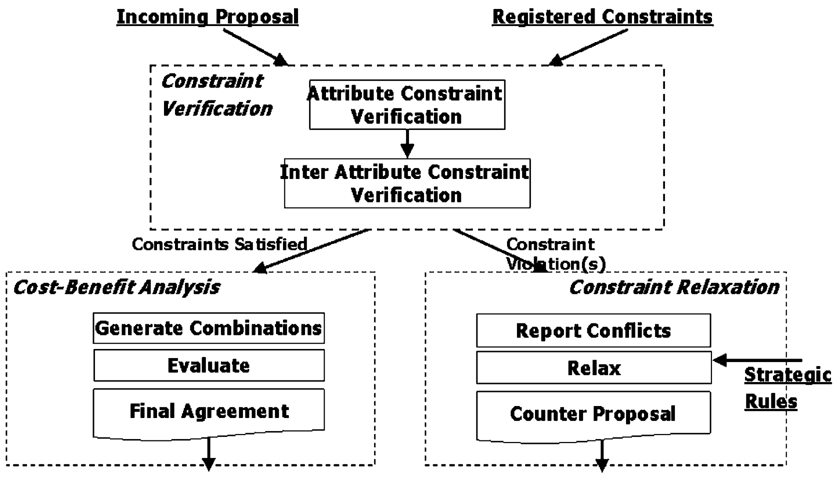


Fig. 2. Proposal processing flow

cording to the priority, the NotNegotiable constraint for quantity is checked first. NSB finds that the user's quantity constraint is satisfied. Then, the delivery_day constraint is checked and NSB finds a violation since the supplier specifies the delivery_day constraint in the range [14..21], whereas the buyer requires a delivery day range of [3..10]. As a result, NSB posts a violation event "SupplierComputer_Systemdelivery_day" and triggers rule SR1 to relax the Supplier's deliveryday constraint. Based on the action part of the rule, NSB changes the supplier's delivery_day constraint from RANGE [14..21] to RANGE [12..21]. No other constraints are violated and the outgoing counterproposal to the buyer is as follows. Note, the outgoing counterproposal does not contain priorities since the buyer uses his own constraint evaluation order.

As illustrated by the negotiation rules of the supplier, the supplier uses a combination of cooperative and competitive negotiation strategies. The same assumption is made on the buyer's side. In this context, "cooperative" means that the negotiator is willing to relax his constraints when constraint violations are detected. "Competitive" means that the negotiator does not want to inform the other parties of his bottom-line values as specified in their negotiation rules. Furthermore, if a bottom-line value is reached and the violation still exists, a proposal will be rejected but the other side will be informed of the reason for the rejection. Thus, it is the other side's turn to decide if a concession should be made. Different negotiation strategies and methods of generating counterproposals are possible. They are the subjects of our on-going research. When the counterproposal arrives at the buyer's NSA, it will go through the same processing procedure except that the registered constraints, strategic rules, and the cost-benefit evaluation model of the buyer are used. After the constraint evaluation, NSA may decide to reject or accept the counterproposal, or generate another counterproposal to be sent back to NSB. This back-and-forth process continues until a mutual agreement is achieved or either side terminates the negotiation process.

5.3.1 Negotiation protocol

In order to ensure effective and meaningful communication between negotiation servers, the servers must follow a well-defined protocol to exchange negotiation primitives and data in a bargaining process. In this paper, we use two state transition diagrams to describe our bilateral bargaining protocol. They are shown in Figs. 3a and b. Figure 3a defines the states and state transitions of the buyer and Fig. 3b defines those of the supplier. Several negotiation primitives shown in Table 1 are not included in these diagrams because their usage is self-explanatory.

There are a total of 15 different negotiation states for a negotiation server during a bilateral bargaining process. Since a negotiation server can be the initiator of a negotiation transaction and, at the same time, the recipient of a transaction initiated by another server, the transition diagrams define the various states in which a server can be in when playing both roles. We shall use "buyer" and "supplier" to represent these two roles. The meanings of the states are shown in Table 2.

S0 is the initial state. S1 is entered after the buyer sends out a CFP. S7 is entered when the supplier receives a CFP. The remaining states are shared by the buyer and the supplier. In general, a negotiation transaction is initiated by the buyer after sending out a CFP. In the CFP, the buyer indicates its interest to start a negotiation on some negotiation item(s) and provides some constraints for the negotiation item(s). To respond

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 Acknowledgement of the Propose/CFP that was sent; request Withdrawal of that Propose/CFP
S12	Received Acknowledgement of the Propose/CFP that was sent; request Modification of that Propose/CFP
A	Agreement is reached
T	Negotiation is terminated unilaterally

```
ENTITYProposal {
   ATTRIBUTE-CONSTRAINT
    model        String      ENUMERATION{PII350, PII400}
    memory       Integer     ENUMERATION {32m,64m, 96m}
    monitor      Integer     ENUMERATION {17, 19}
    hard_drive   Integer     ENUMERATION {4g, 6g, 8g}
    unit_price   Float       DERIVED
    deliver_day  Integer     RANGE[12..21]
    quantity     Integer     RANGE [250..550]
   INTER-ATTRIBUTE-CONSTRAINT
     quantity_deliver_day_1     quantity >= 400 implies  deliver_day >= 16
     model_memory_1             model = 'PII400' implies memory >= 64m
}
```

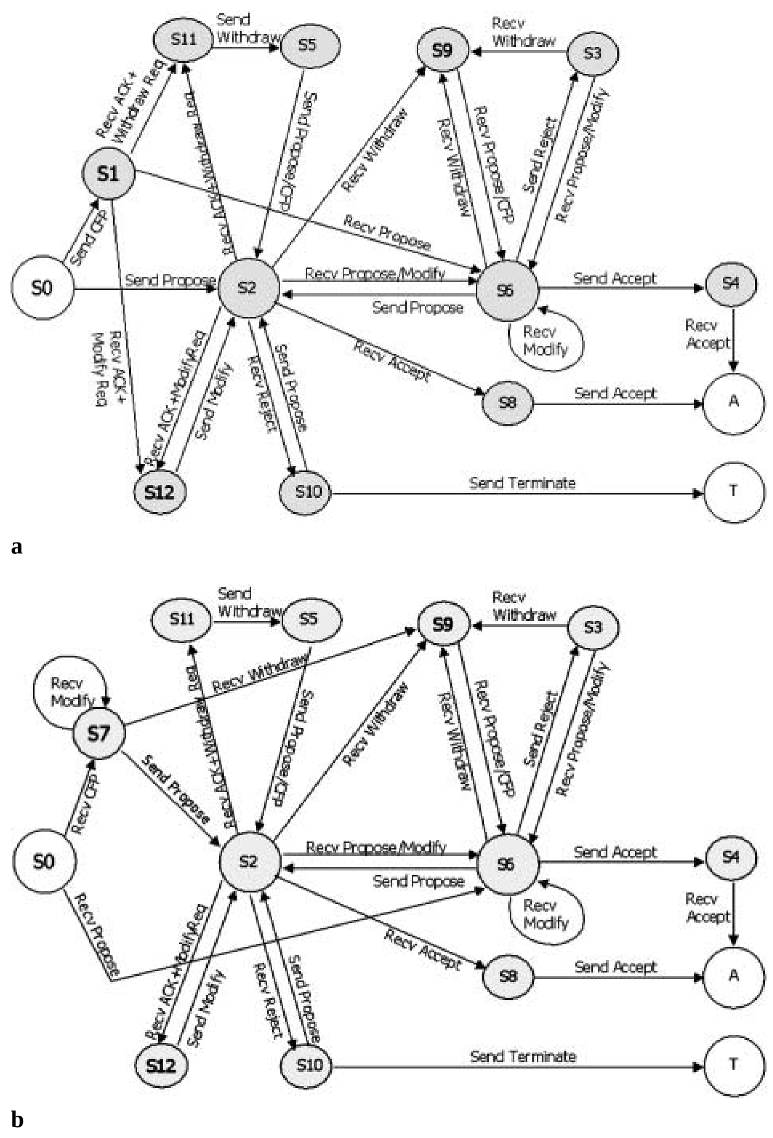


Fig. 3. Buyer's negotiation protocol state transition diagram

to a CFP, the supplier would send a Propose message (containing the 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 message (i.e., counterproposal) 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 CFP. Hence, the supplier may also receive a Propose to start a negotiation. Either case (using CFP or Propose to start the negotiation) is defined by the messages and state transitions surrounding S0.

The primitive Propose is used to transmit a proposal or counterproposal between two negotiation servers. S2 and S6 are the two states of a negotiation server when it sends and

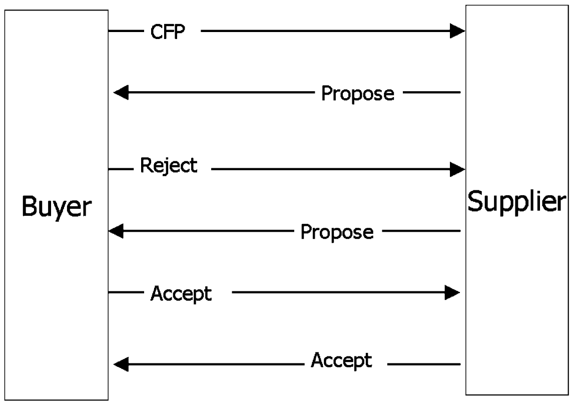


Fig. 4. Counterproposal scenario

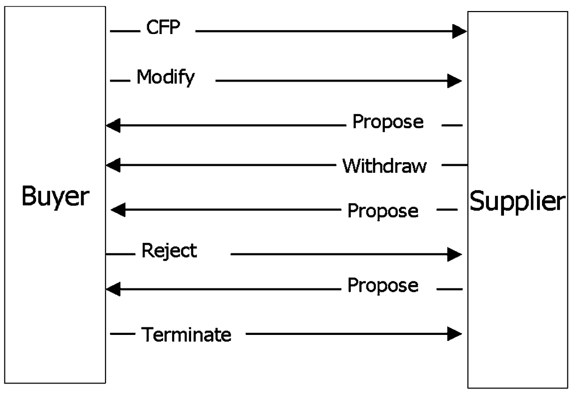


Fig. 5. Withdraw/modify scenario

receives Propose, respectively. To respond to a Propose, several messages are allowed: send out another Propose to issue a counterproposal, send out a Reject to indicate the rejection of some conditions specified in the original proposal, and send out an Accept to indicate the acceptance of the terms and conditions specified. Note, a Reject message is different from a Terminate message. It is used to convey the dissatisfaction of some conditions to the counterpart and the expectation to modify the previous proposal and send the modified one back. Thus, the Reject message contains the rejected data attributes. The Terminate message is used to unilaterally terminate a negotiation process. 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 message is received or sent, the negotiation will move to state T. In order to keep the diagrams readable, we have not shown the state transitions to T.

An agreement in a bilateral negotiation is reached when both sides exchange Accept messages with identical contents. The messages from states S2 to S8 and S8 to A illustrate this case. This is analogous to putting the two signatures of the parties involved in a traditional business transaction on a final agreement.

5.3.2 Negotiation scenarios

To further clarify the semantics and usage of the negotiation primitives and the negotiation protocol, we provide the following negotiation scenarios to show the meaningful state transitions and message exchanges between two sample negotiation servers.

Counterproposal scenario

Figure 4 shows that the buyer starts a CFP (Buyer State: S0→S1) and the supplier returns a proposal (Propose) (Supplier State: S0→S7→S2) to the buyer (Buyer State: S1→S6). After evaluating the proposal, the buyer decides to reject the proposal (Buyer State: S6→S3). After receiving the buyer's Reject (Supplier State: S2→S10), the supplier returns a modified proposal 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's State Transitions: S6→S4→A, Supplier's State Transitions: S2→S8→A).

Scenario with modify/withdraw message exchange

In a negotiation, sometimes a negotiator wants to change some conditions in the previous proposal or cancel the previous proposal that has been sent before the counterpart responds. Our protocol provides two negotiation primitives for this scenario: Modify and Withdraw. When the negotiation server receives a Modify or Withdraw message, it stops the processing of the previous Propose message. In the case of Modify, the negotiation server needs to combine the contents of the previous proposal and the modifications specified in the received Modify message to produce a new proposal. In the case of Withdraw, the negotiation server receives a new proposal from the sender of the Withdraw message.

Both cases require the negotiation server to respond based on the new proposal. As described in the state transition diagram, Modify and Withdraw for the CFP message is similar to the case of Propose. Moreover, since the modification and withdraw of a proposal or CFP generally require human intervention, two human input actions, ModifyReq (for modification request) and WithdrawReq (for withdraw request), are also introduced in the protocol state diagrams. A scenario that involves Modify/Withdraw is shown in Fig. 5.

In this scenario, the buyer sends out a call-for-proposal (CFP) message to the supplier (Buyer State: S0→S1, Supplier State: S0→S7). Before the supplier returns any message, the buyer changes his mind and uses the Modify BOD 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 mind. The supplier then uses the Withdraw message to first notify the buyer of his intention and then sends a new proposal (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 counterproposal (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). We should point out that the sending and receiving of CFP, Propose, Modify, and Withdraw between two parties require synchronization between the two negotiation servers. The handling of these synchronization problems is described in [HUA00].

5.3.3 Negotiation constraint satisfaction processing

As stated above, we regard negotiation proposal evaluation as a constraint satisfaction problem. In traditional CSP algorithms, when the inter-attribute constraints are evaluated, all the constant-value-based combinations that can satisfy the attribute constraints need to be generated and tested against the inter-attribute constraints. This approach may generate a large number of instances. For example, if there are four Integer type negotiation attributes and each attribute constraint defines a range having 1,000 acceptable values, e.g., [1,000 .. 2,000], [5,000 .. 6,000], then the total number of instances generated will be 1,000*1,000*1,000*1,000=1 trillion. Obviously, testing these instances against the inter-attribute constraints is very resource intensive.

To improve the CSP engine performance, we have developed an algorithm that can generate instances based on interval value instead of constant value for each attribute. The interval defines the upper and lower bound values of the attribute. It uses the symbols "(" and ")" to indicate that the boundary values should be exclusive and the symbols "[" and "]" to indicate that the boundary values should be inclusive. All the attribute constraints can be uniformly represented by one or more such intervals. For instance, the constraint "EQUAL a" can be defined as "[a,a]", "ENUMERATION[a, b, c]" can be redefined as "[a,a], [b,b], [c,c]", and "RANGE[a..b]" can be mapped to interval "[a, b]". In this case, after the attribute constraint checking, the result will be a set of intervals for each attribute. We then generate the instances based on the attribute intervals. Since each interval may include multiple constant values, the number of instances generated by this approach would be much less than the one generated by the constant-value-based approach. In order to distinguish the instances of the constant-value-based approach from those instances of the interval-based approach, we call the latter *interval records*.

However, if we generate records immediately after the attribute constraint checking and treat each interval record as a collection of multiple constant-value-based instances, it is possible that some instances of the record may produce a TRUE value with respect to an inter-attribute constraint while the rest of the instances may produce a FALSE value. In this case, we cannot assign a TRUE or FALSE value to the interval record with respect to the inter-attribute constraint. Therefore, we have to split the interval record into two or more separate records so that each new record can have its own evaluation result. We describe the interval splitting algorithm below.

In an inter-attribute constraint, each atomic predicate which contains a relational operator (e.g., =, !=, >, <, $\geq$ or $\leq$) can be represented by one or more attribute intervals. For example, x$\geq$100 can be represented by the interval [100, POSITIVE_INFINITE). To distinguish the attribute intervals derived from an inter-attribute constraint from intervals derived from an attribute constraint, we call the latter an *attribute constraint interval*. In the evaluation, if an attribute constraint interval overlaps with the attribute interval derived from an atomic predicate of an inter-attribute constraint, we split the attribute constraint interval into two or more intervals. For example, an attribute constraint interval for attribute x is Ax. An attribute interval derived from a predicate of x in the inter-attribute constraint is Bx. When checking the inter-attribute constraint, we create two new intervals for the attribute x. The first interval is [Ax $\cap$ Bx], and the other interval is [Ax - (Ax $\cap$ Bx)]. In fact, the latter formula may produce two intervals. The same task of interval splitting needs to be performed for all other attributes that are used in the predicates of the inter-attribute constraint, e.g., attributes y and z. We repeat the above procedure to check the attribute constraint intervals against other inter-attribute constraints. As a result, more new intervals may be produced for each attribute. Finally, we can generate interval records based on all the combinations of new intervals for all the attributes (e.g., x, y, and z). We shall illustrate the process using the following example. Let us assume that the attribute-constraints and inter-attribute constraints after the attribute constraint checking process are:

```
X:  RANGE [10 .. 110]
Y:  RANGE [300 .. 600]
IAC1:  X > 50 → Y > 400
IAC2:  X < 70 → Y < 500
```

In a traditional CSP, a total of 100*300 = 30,000 instances will be generated and tested against the two inter-attribute constraints. In our algorithm, the initial attribute constraint intervals are:

```
X:  [10, 110]
Y:  [300, 600]
```

First, we evaluate the constraint intervals of X and Y against the inter-attribute constraint IAC1 which results in the splitting of each interval into two as follows:

```
X:  [10, 50], (50, 110]
Y:  [300, 400], (400, 600]
```

Next, we evaluate the new intervals against IAC2, resulting in further splitting:

```
X:  [10, 50], (50, 70), [70, 110]
Y:  [300, 400], (400, 500), [500, 600]
```

After we finish the interval splitting process, we can generate interval records based on these new intervals as shown in Table 3.

After generating the records, we can derive the evaluation result for each record against the inter-attribute constraints as

Table 3. Interval-based records

Record number	X	Y	IAC1 Result	IAC2 Result	Satisfied
1	[10..50]	[300..400]	T	T	Yes
2	[10 .. 50]	(400..500)	T	T	Yes
3	[10 ..50]	[500..600]	T	F	No
4	(50..70)	[300..400]	F	T	No
5	(50..70)	(400..500)	T	T	Yes
6	(50..70)	[500..600]	T	F	No
7	[70..110]	[300..400]	F	T	No
8	[70..110]	(400..500)	T	T	Yes
9	[70..110]	[500..600]	T	T	Yes

follows: if the evaluation result for any inter-attribute constraint is FALSE, the record should be removed. Otherwise, the record is kept as part of the answer. For example, as shown in the above table, we can remove records 3, 4, 6, and 7. The remaining records are the records accepted by the CSP engine based on the input attribute and inter-attribute constraints. As we can see, because each interval covers multiple attribute values, the number of generated records is much smaller than the number of constant value instances (nine instead of 30,000). Therefore, the performance of our improved CSP engine is better.

After the inter-attribute constraint evaluation is completed, if a violation of any inter-attribute constraint is found (i.e., no instance satisfies the constraint), an event is posted to activate some strategic rules to either automatically relax the constraint or call for human intervention. If no violation is found (meaning, there exists at least one negotiation item instance that satisfies all inter-attribute constraints), the proposal is acceptable to the client. In that case, an Accept message is sent to the other party. In the event that multiple negotiation item instances satisfy all constraints of the client, the cost-based decision module is invoked to quantitatively evaluate the alternatives. The negotiation item instance most beneficial to the client is selected for acceptance.

6 Quantitative cost-benefit decision model

Negotiation is a complex decision-making process. The decisions may include not only the rejection or acceptance of a proposal, but also the evaluation of and the selection from multiple choices. The latter is needed in the following situations:

- A client receives a proposal, which specifies a number of alternative negotiation conditions (e.g., purchase computer with characteristics a, b, and c or another model with characteristics d, e, f, and g).
- A client may conduct negotiations simultaneously with multiple parties. Different offers need to be evaluated to get their relative cost-benefit ratings in order to do the proper selection.
- A single proposal may contain a large number of value combinations, which need to be evaluated to make the optimal selection when determining the final agreement or when forming a counterproposal. We note here that a proposal uses range, enumeration, and value constraints to specify purchase or sales requirements.
- A number of attribute and inter-attribute constraint violations have been found in a proposal. There are different ways of relaxing the constraints by different combinations of values (e.g., reduce the price of a computer and also reduce the memory and monitor sizes or leave the price as it is and increase the length of the service coverage).

A systematic, quantitative, and justifiable cost-benefit evaluation model is needed for implementing an automated negotiation system. In this work, we have adapted the cost-benefit decision model (CBDM) reported in [SU87]. In our model, the contents of each proposal instance are divided into two structures: (1) the cost structure, which contains all the attributes to which costs can be assigned (e.g., a specific disk, an additional memory board, etc.); and (2) the preference structure, which contains all the attributes to which preference scores can be assigned subjectively (note: these two sets of attributes may overlap). The structures are separately analyzed to obtain two aggregated values: an aggregated cost value and the global preference score. These two values are then combined to derive a global cost-preference indicator for each combination of data conditions in a proposal instance. The preference scoring and aggregation functions associated with all the attributes are specified by negotiation experts during the registration process. Forms accessible through Web browsers are provided for this registration purpose. Costs associated with different values of these attributes can usually be found in an inventory system.

Table 4. Aggregate function spectrum

Minimum	$E = \min(e1, e2, \ldots, en)$	$r = -\infty$
Harmonic mean	$E = 1/(w1/e1 + w2/e2 + \ldots + wn/en)$	$r = -1$
Geometric mean	$E = (e_1)^{w1}.(e_2)^{w2} \ldots$	$r = 0$
Weighted arithmetic mean	$E = w1{*}e1{+}w2{*}e2{+}\ldots wn{*}en$	$r = 1$
Square mean	$E = \sqrt{w_1 * e_1^2 + w_2 * e_2^2 + ...}$	$r = 2$
Maximum	$E = \max(e1, e2, \ldots, en)$	$r = +\infty$

In the preference analysis based on the preference structure, preference scores ranging from zero to one are assigned to all attribute-value pairs using a set of "elementary criteria." An elementary criterion is a mapping from an attribute-value or an attribute-value-range pair to a real number between zero and one. The real value expresses a client's degree of satisfaction for the particular attribute value. For example, if Vendor-Service is an attribute of computer, a client may assign a preference score of 0 if the value of Vendor-Service is "mornings only", 0.3 if it is "daytime-only", and 1 if there is 24-h service. Here, the score of 1 means one hundred percent satisfaction. For attributes whose values cannot be enumerated, e.g., mean-time-to-failure of a disk, evaluation functions can be defined and used as the elementary criteria. The elementary preference scores are weighted and aggregated into a global preference rating using a spectrum of "preference aggregation functions," which are derived from a weighted power mean [DUJ75]:

$$E = (w_1.e_1^r + w_2.e_2^r + ... + w_n.e_n^r)^{1/r}$$

By varying the value of r, a spectrum of aggregation functions is generated, including functions such as *min*, *max*, *weighted arithmetic mean*, etc. Some commonly used functions are given in Table 4.

The aggregation functions represent different degrees of conjunction and disjunction of negotiation data conditions. They can be selected by a user to suit different decision situations and for the selection of different products and services. For example, the maximum aggregation function is suitable when one or more of the negotiation conditions are acceptable to the user. In that case, the maximal value among all the preference scores derived for the negotiation conditions will be used as the global preference score. For example, if CPU speed is most important to a client and he is 90% satisfied (i.e., preference score of 0.9) with the speed of the computer under consideration, the preference scores of the rest of attributes can be ignored. In this case, the global preference score is 0.9. On the other end of the spectrum, the Minimum function would use the minimal score among all the preference scores as the global preference. In the above example, if a client is only 10% satisfied with the speed of the CPU, the global score is 0.1 even though he may be totally satisfied with all other attribute values. As pointed out above, the aggregation functions defined by negotiation experts represent different degrees of conjunction/disjunction. A naïve user, who cannot be expected to know the mathematics behind these functions, can be asked to select a value from the range (0,1) to express his/her desired degree and the system can map the value to the proper aggregation function.

7 Implementation

The research and implementation effort of our Internet-based negotiation server is supported by a NIST project, which focuses on supply chain management using a network of heterogeneous component systems (i.e., application systems, agents, rule-based systems, etc.). Negotiation is a necessary activity among the many components of a supply chain and requires the interchange of data and knowledge among the negotiating parties.

7.1 System architecture and components

The UF Negotiation Server consists of the components shown in Fig. 6. In the following, the term *transaction* is used to refer to the entire negotiation process between a buyer and a supplier (i.e., starting from a Call-for-proposal to the acceptance or termination of a proposal). The term *session* is used to refer to each exchange of a message between two negotiation servers. Hence, each negotiation step is called a *negotiation session*. We now briefly describe the functionality and implementation techniques for each component:

(1) *Register web page* is a Web-based graphical user interface (GUI) for registering negotiation knowledge, including requirements, constraints, negotiation strategies, and data to be used in the cost-benefit evaluation model. A snapshot of a requirements registration session is shown in Fig. 7. In this figure, we see the requirements for a seller of EtherNetCards in terms of attribute and inter-attribute constraints. For example, we can see that the seller would like to sell between 250 and 550 cards (quantity constraint) within 2 to 3 weeks after the closing of the deal (delivery_period constraint). The sole inter-attribute constraint specifies that if the quantity reaches 400 cards, then the delivery_period must be at least 16 days after the closing of the deal.

(2) *Registration* is a Java servlet that dynamically generates the registration Web pages according to the metadata description of the negotiation item provided by an ontology service, and stores the user registration information in the repository.

(3) *Repository* provides persistent data service for the negotiation server. It is RMI-based and consists of two modules: MetaDataManager for storing and managing the ontology information for each negotiated item and NegotiationInformationBase for storing and managing the negotiation knowledge gathered during registration and at runtime.

(4) *Client monitor* is a Web-based graphical user interface (GUI) which uses Java applets for monitoring the ongoing negotiation. The information presented includes output messages related to constraint checking, rule triggering, and final acceptance of a proposal or counterproposal. Figure 8 shows our client monitor displaying a snapshot of an ongoing negotiation.

(5) *Client notification* is a socket-based communication channel between the negotiation server and the Client Monitor.

(6) *Cost benefit module* provides cost-benefit analysis and evaluates negotiation alternatives as explained in Sect. 6. It provides APIs for different services, including returning the best alternative, returning all the alternatives with preference scores, etc.

(7) *Constraint satisfaction processing* is responsible for evaluating the constraints provided in a negotiation proposal or counterproposal against the pre-registered constraints in the server. It identifies the constraints that have been violated and posts events to trigger the processing of negotiation strategic/tactic rule(s) as explained in Sect. 5.3.1.

(8) *ETR Server* is the event-trigger-rule server responsible for receiving events from the negotiation manager and triggers the proper strategic/tactic rules to relax constraints or initiate human intervention. The use of ETR rules to implement negotiation strategies was described in Sects. 4.2 and 5.3.1 and the implementation of the ETR server is described in [LAM98, LEE00].

(9) *Messaging* is our Internet-based communication interface with other negotiation servers. A set of general communication interfaces is defined for all of our upper level programs to provide support for different communication protocols.

(10) *XML BOD Processing* wraps and unwraps messages for communication between negotiation servers. An important feature of our system architecture is the interoperability among different negotiation systems. One approach to providing interchange among heterogeneous components is by introducing a standard describing how to represent and format the information, and requiring that all parties adhere to it. For this project, we have adapted the Open Applications Group's (OAG's) business object documents (BODs) [OPE95], which are wrapped in XML [W3C98] as the standard interchange format. In the OAG specification, each BOD has a verb and a noun. The verb specifies a business operation and the noun specifies the name of a pre-defined data structure that contains the data that is transmitted as part of the operation. In our implementation, we have defined a set of negotiation BODs, one for each negotiation primitive shown in Table 1 of Sect. 4.1. Following OAG's recommendation, the negotiation primitive forms the verb and the word "proposal" is used as the noun: for example, Reject-Proposal. An XML parser in each negotiation server parses each incoming XML-BOD and extracts the data from it. The contents of the BOD are converted into internal objects for further processing.

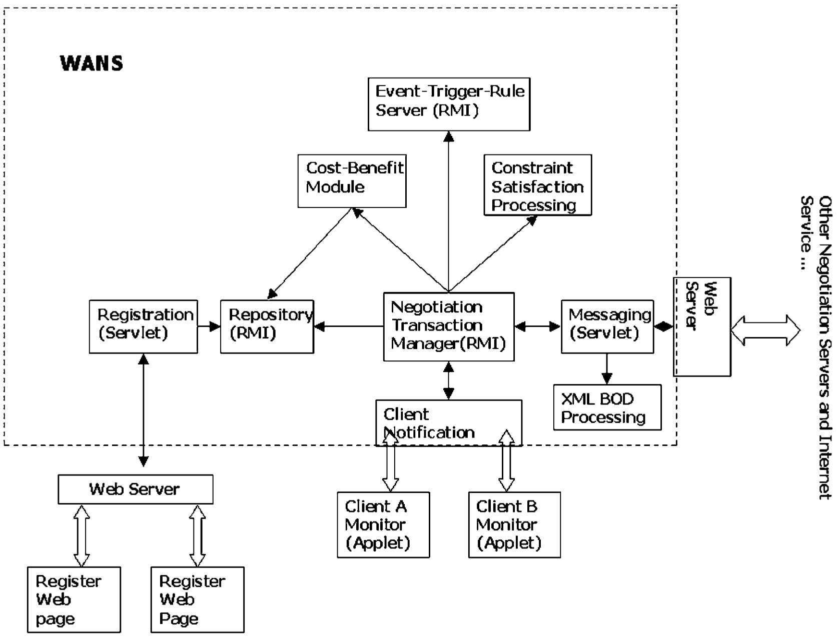
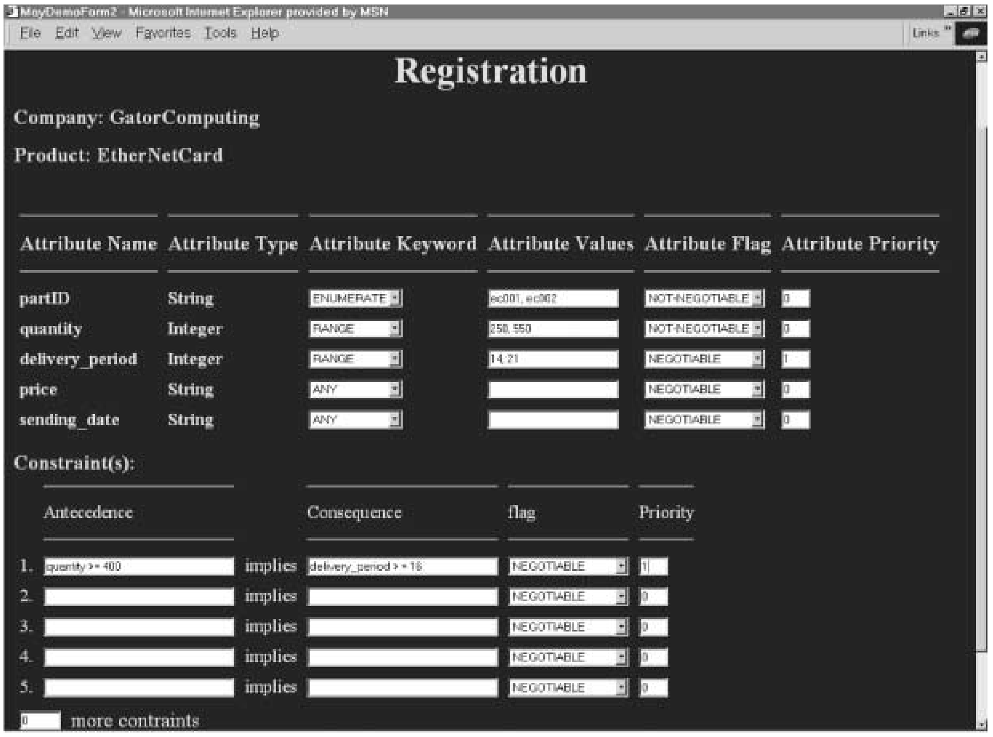

WANS
Event-Trigger-Rule Server (RMI)
Cost-Benefit Module
Constraint Satisfaction Processing
Registration (Servlet)
Repository (RMI)
Negotiation Transaction Manager(RMI)
Messaging (Servlet)
Web Server
XML BOD Processing
Client Notification
Web Server
Client A Monitor (Applet)
Client B Monitor (Applet)
Register Web page
Register Web Page
Other Negotiation Servers and Internet Service ...

Fig. 6. System architecture



MayDemoForm2 - Microsoft Internet Explorer provided by MSN

Registration

Company: GatorComputing

Product: EtherNetCard

Attribute Name	Attribute Type	Attribute Keyword	Attribute Values	Attribute Flag	Attribute Priority
partID	String	ENUMERATE	ec001, ec002	NOT-NEGOTIABLE	0
quantity	Integer	RANGE	250, 550	NOT-NEGOTIABLE	0
delivery_period	Integer	RANGE	14, 21	NEGOTIABLE	1
price	String	ANY		NEGOTIABLE	0
sending_date	String	ANY		NEGOTIABLE	0

Constraint(s):

	Antecedence		Consequence	flag	Priority
1.	quantity >= 400	implies	delivery_period > = 16	NEGOTIABLE	1
2.		implies		NEGOTIABLE	0
3.		implies		NEGOTIABLE	0
4.		implies		NEGOTIABLE	0
5.		implies		NEGOTIABLE	0

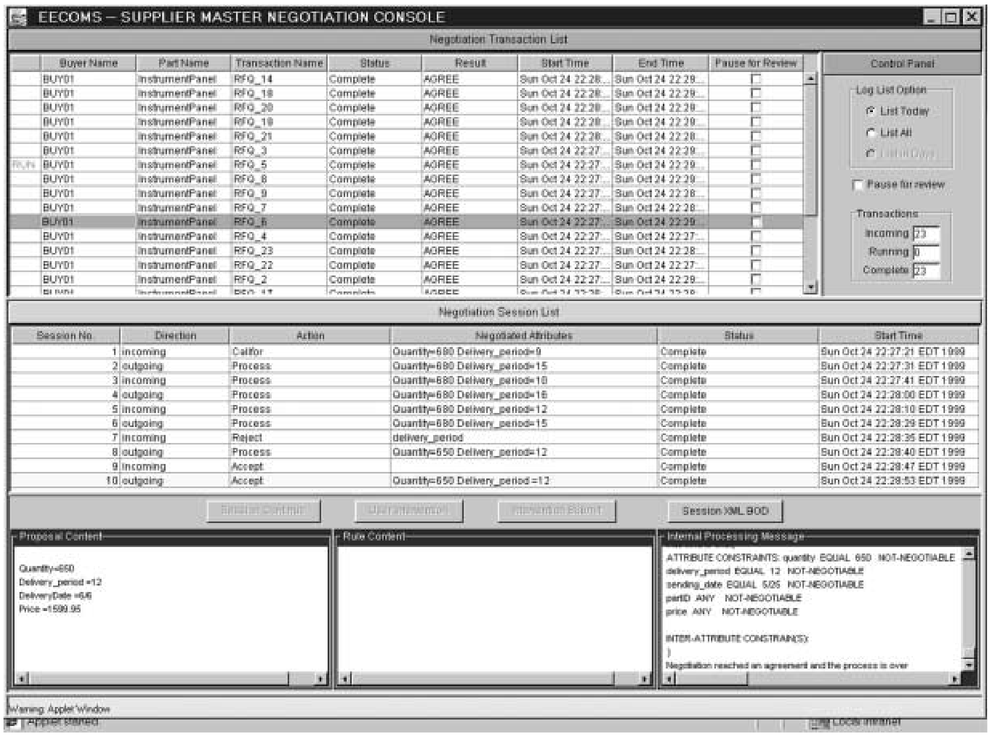
0 more contraints

Fig. 7. Snapshot of the web-based registration interface

(11) *Negotiation transaction manager* is a server program which supports Java's RMI and manages multi-user and multi-thread negotiation transactions. Following the states and transitions of the negotiation protocol, it coordinates all the different components to complete the negotiation tasks automatically. Presently, it implements the bilateral bargaining protocol.

7.2 Integrated testing

We have integrated two copies of our negotiation server, its supporting tools and GUIs, with an existing supply chain. The supply chain consists of a buyer and multiple suppliers and manufacturers together with their supporting infrastructure for secure communication, decision-making, and supplier selection. One of the suppliers was represented by a commercial ERP system, others participated in a more loosely connected fashion in the form of negotiation-enabled Web servers. The goal of this integration was to demonstrate the acquisition of and the negotiation for Ethernet cards with multiple suppliers in a supply chain application. We briefly describe the control and data flow in the supply chain to illustrate the negotiation scenario.


EECOMS – SUPPLIER MASTER NEGOTIATION CONSOLE

Negotiation Transaction List

	Buyer Name	Part Name	Transaction Name	Status	Result	Start Time	End Time	Pause for Review
	BUY01	InstrumentPanel	RFQ_14	Complete	AGREE	Sun Oct 24 22:28:...	Sun Oct 24 22:29:...	☐
	BUY01	InstrumentPanel	RFQ_18	Complete	AGREE	Sun Oct 24 22:28:...	Sun Oct 24 22:29:...	☐
	BUY01	InstrumentPanel	RFQ_20	Complete	AGREE	Sun Oct 24 22:28:...	Sun Oct 24 22:29:...	☐
	BUY01	InstrumentPanel	RFQ_19	Complete	AGREE	Sun Oct 24 22:28:...	Sun Oct 24 22:29:...	☐
	BUY01	InstrumentPanel	RFQ_21	Complete	AGREE	Sun Oct 24 22:28:...	Sun Oct 24 22:28:...	☐
	BUY01	InstrumentPanel	RFQ_3	Complete	AGREE	Sun Oct 24 22:27:...	Sun Oct 24 22:29:...	☐
RUN	BUY01	InstrumentPanel	RFQ_5	Complete	AGREE	Sun Oct 24 22:27:...	Sun Oct 24 22:29:...	☐
	BUY01	InstrumentPanel	RFQ_8	Complete	AGREE	Sun Oct 24 22:27:...	Sun Oct 24 22:29:...	☐
	BUY01	InstrumentPanel	RFQ_9	Complete	AGREE	Sun Oct 24 22:27:...	Sun Oct 24 22:28:...	☐
	BUY01	InstrumentPanel	RFQ_7	Complete	AGREE	Sun Oct 24 22:27:...	Sun Oct 24 22:28:...	☐
	BUY01	InstrumentPanel	RFQ_6	Complete	AGREE	Sun Oct 24 22:27:...	Sun Oct 24 22:29:...	☐
	BUY01	InstrumentPanel	RFQ_4	Complete	AGREE	Sun Oct 24 22:27:...	Sun Oct 24 22:27:...	☐
	BUY01	InstrumentPanel	RFQ_23	Complete	AGREE	Sun Oct 24 22:27:...	Sun Oct 24 22:28:...	☐
	BUY01	InstrumentPanel	RFQ_22	Complete	AGREE	Sun Oct 24 22:27:...	Sun Oct 24 22:27:...	☐
	BUY01	InstrumentPanel	RFQ_2	Complete	AGREE	Sun Oct 24 22:27:...	Sun Oct 24 22:29:...	☐

Control Panel

Log List Option: [✓ List Today] List All

☐ Pause for review

Transactions: Incoming 23; Running 0; Complete 23

Negotiation Session List

Session No.	Direction	Action	Negotiated Attributes	Status	Start Time
1	incoming	Callfor	Quantity=680 Delivery_period=9	Complete	Sun Oct 24 22:27:21 EDT 1999
2	outgoing	Process	Quantity=680 Delivery_period=15	Complete	Sun Oct 24 22:27:31 EDT 1999
3	incoming	Process	Quantity=680 Delivery_period=10	Complete	Sun Oct 24 22:27:41 EDT 1999
4	outgoing	Process	Quantity=680 Delivery_period=16	Complete	Sun Oct 24 22:28:00 EDT 1999
5	incoming	Process	Quantity=680 Delivery_period=12	Complete	Sun Oct 24 22:28:10 EDT 1999
6	outgoing	Process	Quantity=680 Delivery_period=15	Complete	Sun Oct 24 22:28:29 EDT 1999
7	incoming	Reject	delivery_period	Complete	Sun Oct 24 22:28:35 EDT 1999
8	outgoing	Process	Quantity=650 Delivery_period=12	Complete	Sun Oct 24 22:28:40 EDT 1999
9	incoming	Accept		Complete	Sun Oct 24 22:28:47 EDT 1999
10	outgoing	Accept	Quantity=650 Delivery_period =12	Complete	Sun Oct 24 22:28:53 EDT 1999

Proposal Content

```
Quantity=650
Delivery_period =12
DeliveryDate =6/6
Price =1599.95
```

Rule Content

Internal Processing Message

```
ATTRIBUTE CONSTRAINTS: quantity EQUAL 650 NOT-NEGOTIABLE
delivery_period EQUAL 12 NOT-NEGOTIABLE
sending_date EQUAL 5/26 NOT-NEGOTIABLE
partID ANY NOT-NEGOTIABLE
price ANY NOT-NEGOTIABLE

INTER-ATTRIBUTE CONSTRAIN(S):
}
Negotiation reached an agreement and the process is over
```

Warning: Applet Window

Applet started

Local intranet

Fig. 8. Session monitor displaying a snapshot of an ongoing negotiation

The commercial requisition system on the buyer side initiates a requisition by sending an Add-Requisition BOD to the Supplier Selection Component through a messaging infrastructure. The latter makes use of the information it maintains on regular suppliers (the suppliers that the buyer usually does business with) to come up with a purchasing plan, which specifies how many units of the ordered item should be ordered from which supplier and at which delivery date. A number of (supplier, quantity, delivery-date) triplets are generated to meet the demand of the buyer. In the demonstration scenario, the Supplier Selection Component issues a ProcessRFQ BOD (Process Request-for-Quote), which contains one of the triplets, to the Negotiation Server that represents the buyer. The negotiation server generates the Call-for-Proposal BOD and starts the negotiation process by sending the call-for-proposal to the negotiation server of the supplier side.

After possibly several rounds of exchanging counterproposals, the quantity and delivery date of the original triplet may be changed due to the supplier's constraints and rules. The price information associated with the negotiated result is retrieved from the supplier's inventory database. The negotiated result and price information are returned to the Supplier Selection component using the AckRFQ BOD (Acknowledge-Request-for-Quote). Three scenarios are possible: (1) the supplier accepts the order by providing its proposed delivery schedule; (2) the supplier returns a counter offer with a new delivery schedule and quantities; and (3) the supplier rejects the order. In case of a reject, it is possible that the Supplier Selection Component may have to contact additional suppliers to get the number of items the buyer needs. Once the negotiations are completed, the Supplier Selection component sends the AddPO BOD (Add-Purchase-Order) to the requisition system to initiate the order. There are several other component systems involved in the integrated demo. We have not described them here because they are not directly related to our discussion of the negotiation system.

8 Conclusion and discussion

In this paper, we have presented the design and implementation of a replicable Internet-based negotiation server for conducting bargaining-type negotiations. Our system is intended for use in e-commerce and e-business applications. A content specification language for information registration, a negotiation protocol and its primitive operations, an automated negotiation process, a cost-benefit decision model, and the architecture of an implemented system have been described. In contrast to most of the existing work on negotiation, which is based on a distributed agent technology, our work makes use of object-oriented, active database technology [WID96]. For example, our content specification language is based on an extended object model. The language and its accompanying GUI tools are used by buyer companies to specify the data characteristics and constraints of goods and services in which they are interested in, and by seller companies to publish information about the goods and services they provide. Negotiation experts use this language and our GUI tools to specify negotiation strategies in terms of active negotiation rules and to provide the information for cost-benefit analysis and selection. The same language is used by clients to define proposals and counterproposals, which are wrapped in XML BODs and are transmitted between negotiation servers through a messaging infrastructure. The constraint, event, rule, and trigger specification facilities of the language and its object model are extensions of the traditional object definition language and model, in which objects are defined only in terms of attributes and methods. The added semantics in the specifications of goods and services and negotiation proposals and counterproposals allow negotiation servers to carry out constraint verification, activation of negotiation strategic rules, and generation of counterproposals in an automated fashion.

In our research, we have found that existing work on auctions and bargaining type negotiations focuses mainly on single negotiation items (e.g., price alone) and uses a single elementary scoring function and a single aggregation function (e.g., [SIE97]) to evaluate proposals. In addition, several projects on negotiation make use of the constraint satisfaction programming (CSP) technique to determine if some given data can be generated by or violate a set of constraints. However, such systems are not able to determine which specific constraint is violated when a violation is found. Thus, they are not able to apply the proper rule to relax the constraint. In distinction, our work tackles negotiation problems that involve multiple negotiation items and uses a number of elementary scoring criteria and a spectrum of aggregation functions to conduct cost-benefit analysis. We combine the strengths of a Constraint Satisfaction Process and an Event-Trigger-Rule Sever to evaluate constraints in a priority order, post proper violation events, and trigger proper strategic rules to relax the violated constraints or to take other actions.

We have integrated two copies of the implemented server with several supply chain management components provided by other industrial and university members of a consortium (http://www.ciimplex.org). The four negotiation services described in Sect. 2 have been demonstrated to industrial companies, universities, and government organizations in several public demonstrations including one at the IEEE's Data Engineering Conference [HAM00]. All these services can be accessed through the Web without any client software installation and configuration. A Web-based run-time monitoring and human intervention facility is also implemented using applet and servlet technologies. We make use of an existing object manager to provide the metadata management and persistence support for the registration service. The Event-Trigger-Rule server used in this work is a modified, Web-enabled version of an existing server, which was implemented to run in a CORBA-based environment.

Our current R& D efforts include:

- Investigate different negotiation policies and strategies and their implementation, using our negotiation system as the test platform.
- Include temporal constraints into our constraint language.
- Exchange constraints and rules between heterogeneous rule-based systems through a constraint interlingua [GRO99].

References

[BEA96] Beam C., Segev A., Shanthikumar J.G.: Electronic negotiation through Internet-based auctions. Technical Report, University of California at Berkeley, 1996

[BEA97] Beam C., Segev A., et al.: Electronic catalogs and negotiations. Technical Report, University of California at Berkeley, 1997

[BIN99] Binmore K., Vulkan N.: Applying game theory to automated negotiation. In: Netnomics 1(1), 1999

[CHA96] Chaves A., et al.: An agent marketplace for buying and selling goods. In: Proc. 1st International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology. London, UK, 1996

[CHA97] Chaves A., et al.: A real-life experiment in creating an agent marketplace. Proc. 2nd International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology. London, UK, 1997

[DAV80] Smith R.G.: The contract net protocol: high-level communication and control in a distributed problem solver. In: Bond A., Gasser L., (eds) Readings in distributed artificial intelligence. Morgan Kaufmann, San Mateo, Calif., USA, 1980

[DUJ75] Dujmovic J.J.: Extended continuous logic and the theory of complex criteria. J. University of Belgrade, Series on Mathematics and Physics, 537: 197–216, 1975

[DWO96] Dworman G., Kimbrough S., Laing J.: On automated discovery of models using genetic programming in game-theoretic contexts. J Manage Inf Syst 12(3), 1996. (http://opim.wharton.upenn.edu/∼dworman/my-papers/HICSSGP6.ps)

[FOX94] Fox M.S., Gruninger M.,: Ontologies for enterprise integration. In: Proc. 2nd International Conference on Cooperative Information Systems (CoopIS). Toronto, Canada, pp. 82–89, 1994

[FIP97] FIPA 97 Specification Version 2.0, part 2, Agent Communication Language, 1997. http://www.fipa.org/spec/fipa97.htm

[GRO99] Grosof B.N., Labrou Y.: An approach to using XML and a rule-based content language with an agent communication language. In: Proc. IJCAI-99 Workshop on Agent Communication Languages (ACL-99), Stockholm, Sweden, Aug. 1, 1999

[HAM00] Hammer J., Huang C., Huang Y.H., Pluempitiwiriyawej C., Lee M., Li H., Wang L., Liu Y., Su S.Y.W.: The IDEAL approach to Internet-based negotiation for e-commerce. System Demonstration at the International Conference on Data Engineering, San Diego, Calif., USA. Feb. 28–March 3, 2000

[HAG00] HaggleWare: www.haggleware.com

[HUA00] Huang C.: A Web-based negotiation server for supporting electronic commerce. Ph.D. Dissertation, Department of Computer and Information Science and Engineering, University of Florida, 2000. http://www.cise.ufl.edu/tech-reports/tech-reports/tr99-abstracts.shtml, TR 010

[JEL89] Jelassi M.T., Foroughi A.: Negotiation support systems: an overview of design issues and existing software. Decis Support Syst 5(2): 167–181, 1989

[JON80] Jones A.J.: Game theory: mathematical models of conflict. Ellis Horwood, Chichester, UK, 1980

[KAR93] Karrass C.L.: Give and take: the complete guide to negotiating strategies and tactics. HarperCollins, New York, 1993

[KAR97] Karacapilidis N.I., Pappis C.P.: A framework for group decision support systems: combining AI tools and OR techniques. Eur J Oper Res 103, 1997

[KAS99] Kasbah URL: http://kasbah.media.mit.edu

[KER99] Kersten G.E., Szpakowicz S.: Modelling business negotiations for electronic commerce. Technical report, 1999. International Institute for Applied Systems Analysis A-2361 Laxenburg, Austria. http://www.iiasa.ac.at

[KUM98] Kumar M., Feldman S.I.: Business negotiation on the Internet. IBM Institute for Advanced Commerce

(IAC) Report, 1998 http://www.ibm.com/iac/reports-technical/reports-bus-neg-internet.html

[LAX86] Lax D.A., Sebenius J.K.: The manager as negotiator: bargaining for cooperation and competitive gain. The Free Press, New York, 1986

[LIM93] Lim L., Benbasat I.: A theoretical perspective of negotiation support systems. J Manage Inf Syst 9(3), 1993

[LAM98] Lam H., Su S.Y.W.: Component interoperability in a virtual enterprise using events/triggers/rules. In: Proc. of OOPSLA '98 Workshop on Objects, Components, and Virtual Enterprise. Oct. 18–22, 1998, Vancouver, B.C., Canada

[LEE97] Lee J.G., et al.: ICOMA: an open infrastructure for agent-based intelligent electronic commerce on the Internet. In: Proc. International Conference on Parallel and Distributed Systems (CPADS). IEEE Computer Society, Los Alamitos, Calif., pp 648–655, 1997. http://seopo.skku.ac.kr/∼feb01/papers/icpads.ps

[LEE00] Lee M.: Event and rule services for achieving a web-based knowledge network. Ph.D. Dissertation, Department of Computer and Information Science and Engineering, University of Florida, April, 2000. http://www.cise.ufl.edu/tech-reports/tech-reports/tr00-abstracts.shtml, TR 002

[MAK00] MakeUsAnOffer: www.makeusanoffer.com

[MAT91] Matwin S., Szapiro T., Haigh K.: Genetic algorithms approach to a negotiation support system. IEEE Trans Syst Man Cybern 21(1): 102–114, 1991

[MUL96] Muller H.J.: Negotiation principles. In: O'Hare G.M.P., Jennings N.R., (eds), Foundations of distributed artificial intelligence (ch. 7). Wiley, New York, 1996

[OHR96] O'Hare G.M.P., Jennings N.R., (eds): Foundations of distributed artificial intelligence. Wiley, New York, 1996

[OLI97] Oliver J.R.: A machine learning approach to automated negotiation and prospects for electronic commerce. J Manage Inf Syst 13(10): 83–112. http://opim.wharton.upenn.edu/∼oliver27/papers/jmis.ps

[ONE00] One Accord Technologies, http: www.peacesummit.com/index.html

[OPE95] Open Applications Group Integration Specification, http://www.openapplications.org

[PER00] Perfect: www.perfect.com

[PRO00] Program On Negotiation: www.pon.harvard.edu

[PRI00] Priceline: www.priceline.com

[PRU81] Pruitt D.G.: Negotiation behavior. Academic, New York, 1981

[RAG99] Raghavan H.: Event history processing in an active distributed objects system. Master Thesis, Department of Computer and Information Science and Engineering, University of Florida, Aug. 1999. http://www.cise.ufl.edu/tech-reports/tech-reports/tr00-abstracts.shtml

[RAI82] Raiffa H.: The art and science of negotiation. Belknap, Harvard University, Cambridge, Mass., USA, 1982

[RAN97] Rangaswamy A., Shell G.R.: Using computers to realize joint gains in negotiations: toward an electronic bargaining table. Manage Sci 43(8), 1997

[SAN96] Sandholm T.: Negotiation among self-interested computationally limited agents. Ph.D. Dissertation, Department of Computer Science, University of Massachusetts at Amherst, 1996

[SHE99] Shell G.R.: Open and making concessions. Bargaining for advantage: negotiation strategies for reasonable people (ch. 9). Viking, New York, pp. 156–176, 1999

[SIE97] Sierra C., Faratin P., Jennings N.: A service-oriented negotiation model between autonomous agents. In: Proc. 8th European Workshop on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW-97). Ronneby, Sweden, pp. 17–35

[STA00] Stanford Center on Conflict and Negotiation: www.stanford.edu/group/sccn

[SU87] Su S.Y.W., Dujmovic J., Batory D.S., Navathe S.B., Elnicki R.: A cost-benefit decision model: analysis, comparison, and selection of database management systems. ACM Trans Database Syst 12(3): 472–520, 1987

[SU00] Su S.Y.W., Huang C., Hammer J.: A replicable web-based negotiation server for e-commerce. Proc. Hawaii International Conference on Systems Sciences, Maui, Hawaii, January 4–7, 2000

[W3C98] Word Wide Web Consortium: eXtensibe Markup Language. http://www.w3.org/XML, 1998

[WID96] Widom W. (ed): Active database systems: triggers and rules for advanced database processing. Morgan Kaufmann, San Francisco, Calif., USA, 1996

[WIN00] Win Square: www.winxwin.com

[WOO90] Woo C., Chang M.: An approach to facilitate the automation of semistructured and recurring negotiations in organizations. J Organ Comput 2(1): 47–76, 1990

[YAN00] Yan Y., Yen J., Bui T.X.: A multi-agent based negotiation support system for distributed transmission cost allocation. In: Proc. of the 33rd Annual Hawaii International Conference on System Sciences, Maui, Hawaii, January 4–7, 2000

[ZEN98] Zeng D., Sycara K.: Bayesian leaning in negotiation. Int J Human Comput Syst 48: 125–141, 1998

[ZLO96] Zlotkin G., Jeffrey S., Rosenschein A.: Mechanism design for automated negotiation and its application to task-oriented domains. Artif Intell 86: 195–244, 1996