Automated E-business Negotiation: Model, Life Cycle and System Architecture

Haifei Li, Stanley Y. W. Su, Herman Lam, Yihua Huang

Database Systems Research and Development Center
University of Florida
CSE 470, P.O. Box 116125
Gainesville, FL 32611-6125
{hli, su, hlam, yihuang}@cise.ufl.edu

UF-CISE TR01-005

Abstract

How to apply negotiation principles to E-business is an important topic for both negotiation research and E-business research. Automation of E-business negotiation is even more challenging due to the inherent complexity of business negotiations. Some research has been done in this area, but a comprehensive model for automated E-business negotiation is still missing. Furthermore, existing work in this area does not consider the negotiation process from a full life cycle perspective; therefore valuable information from a previous negotiation is not properly used for the future negotiations. This report discusses two important issues related to automated E-business negotiation: model and life cycle. A system architecture based on the model and the life cycle is proposed.

The negotiation model captures the key concepts and elements involved in automated negotiations. Since the negotiation model is an abstract of an automated negotiation system that implements the business negotiation process using computer networks, it is natural for the model to include different negotiation roles played by different people. Moreover, the model needs to formalize some human activities usually only kept in negotiators' mind. Although our negotiation model includes six aspects of negotiation, this report focuses on two important aspects: the negotiation decision model and negotiation messages. The negotiation decision model captures concepts such as negotiation goals, policies, strategies, plans of decisions and actions, and their inter-relationships. Negotiation messages define a comprehensive set of structured messages to be exchanged among two negotiation systems.

The negotiation life cycle model divides the whole negotiation process into four phases: analysis, design, execution and post-negotiation analysis. The results from the upstream phases are used as inputs into the downstream phases. Since business negotiation is an iterative and continuous process, a feedback mechanism from the post-negotiation analysis phase to previous phases is included. The life cycle model presented in this report covers life cycle models we have surveyed. A system architecture based on the negotiation model and the negotiation life cycle model is proposed and implemented.

1. Introduction

1.1 Motivation and Scope

In the business world, business deals are often made through some kind of negotiation. In years past, negotiations were human-based. They were carried out by representatives of businesses through face-to-face discussions or by using letters, faxes, and/or telephones. In this Internet age, email has become another effective way of human-based negotiation. Human-based negotiation can be very costly, lengthy and error-prone. It would be beneficial if part of or the entire negotiation process can be carried automatically over the Internet as a part of e-business activities. When necessary, humans will be called upon to resolve problems that can not be resolved by an automated system.

Automation of business activities is no longer new. We have witnessed the automation of some business activities with great success. Catalog management and order management are two exemplary business activities that have moved rapidly into cyberspace. Electronic catalogs have many advantages over printed catalogs: namely, reduced cost for printing and distribution, and reduced time to reach potential customers. Moreover, electronic catalogs enable a company to dynamically price its products and modify product specifications to meet customers' needs. Order management software packages accept the orders electronically. These packages enable the company to reduce processing costs, processing error rate, and processing time to fulfill orders. We believe that electronic negotiation is another essential business activity needed to support e-business. It has a bright future, but much work needs to be done.

Some negotiations are conducted with the involvement of a third party, such as peace negotiations (through a third country) and settlements of legal disputes (through an arbitrator). However, business negotiations are usually done bilaterally. For example, when GM decides to purchase computers from IBM, a purchase manager from GM negotiates with a sales manager from IBM without outsiders. If GM is interested in purchasing multiple products from multiple suppliers to make its product, a number of bilateral negotiations with these suppliers may be involved. The focus of our research is therefore on automated bilateral negotiations in which structured messages, instead of free text messages, are exchanged between replicated negotiation servers and processed by them on behalf of the users or business organizations that these servers represent. The approach taken in this work to conduct a bargaining type of negotiation is very general. It can be applied to conduct all sorts of negotiations for products, services, plans, schedules, etc., as long as the subjects of negotiations can be specified by multiple issues (i.e., attributes) and constraints. For example, two business parties can negotiate on price, quantity, delivery date, payment method, and schedule, etc., associated with the purchase of a product or time, quality, payment method, and schedule, etc., of a service.

1.2 Negotiation in E-commerce

Automated negotiation has become an important topic in the e-commerce community. Maes of MIT Media Lab puts negotiation at the center of Consumer Buying Behavior (CBB) model for e-commerce [MAE99]. The model identifies six stages in the buying process: (1) need identification, (2) product brokering, which determines what to buy, (3) merchant brokering, which determines who to buy from, (4) negotiation, (5) purchase and delivery, and (6) product service and evaluation. Nissen of the Naval Postgraduate School has a similar observation about the importance of negotiation in commerce. Nissen proposed a commerce model in [NIS97]. It is shown in Figure 1. Although Nissen does not use the term "negotiation" in his model, it is apparent that "arrange terms" in the model is equivalent to "negotiation". The fact that "arrange terms" is at the central position of the model shows its importance in commerce. Since e-commerce is nothing but commerce over the electronic communications channel, the above model is equally applicable to e-commerce. Feldman, Director of the IBM Institute for Advanced Commerce, also treats negotiation as an important link in the e-commerce value chain [FEL98], as shown in Figure 2. "Consumer" in the value chain diagram does not necessarily mean "individual customer". It can be another business that makes use of the products and services of the producer.

1.3 Deficiencies in Present R&D Efforts on E-business Negotiation

Based on our investigation of the existing academic projects and commercial systems (to be surveyed in Section 2), we have identified two main deficiencies in the current R&D efforts on e-business negotiation. The first deficiency is the lack of a formal model to capture the key concepts and elements involved in automated e-business negotiation. In particular, there is a need for a model which can capture the relationship between business conditions, negotiation goals, policies, strategies, and decision-action rules that drive an automated negotiation system. Most implemented systems, including the current EECOMS's negotiation server, implement some decisions-actions rules (called strategies or tactics in some literature) without showing how these low-level rules relate to high-level business goals and policies. Decision-action rules are tactic rules, which govern how negotiation systems (or human negotiators) behave under certain circumstances. Rules for determining the rejection of a negotiation proposal, the amount of concession, the acceptance of a proposal, etc., are some examples. They are defined by people who play the role of the Negotiation Expert. Whereas, negotiation policies are more general business rules, which are defined by people who play the role of the Policy Maker for achieving some specific business goals. Negotiation policies reflect the Policy Maker's insight in business. It is the responsibility of the Policy Maker to set reasonable goals and define suitable policies for different business conditions and situations. For example, the Policy Maker may set the goal and the policy for negotiating with a Fortune 500 company which are different from those set for negotiating with a one-person company. After the goals and policies are set, the Negotiation Expert can then determine what strategies and decision-action rules to apply in order to implement the policies and to achieve the defined goals. Since the choice of negotiation strategies is controlled by people who play the role of the Negotiation Expert, the Policy Maker does not have to deal with low-level details and tactics of negotiation. On the other hand, the Negotiation Expert has the flexibility to apply his/her domain expertise to conduct negotiations. It is

important to show how changes in business conditions and environments affect the selection of negotiation policies and how policies affect the selection of alternative strategies and decision-action rules used in an automated negotiation system. A formal decision model to capture and relate these concepts is needed.

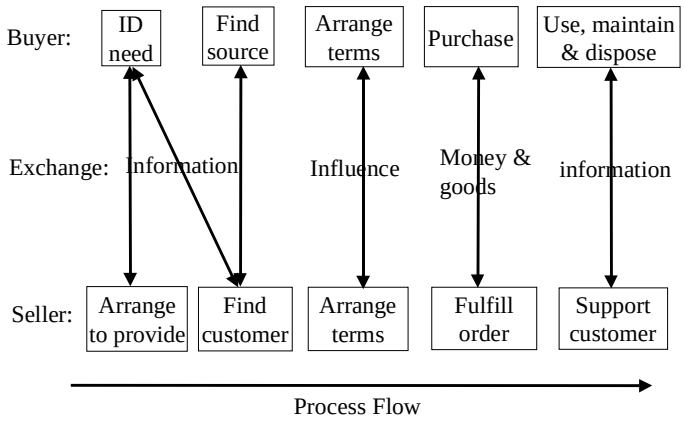

Figure 1. Nissen's Commerce Model

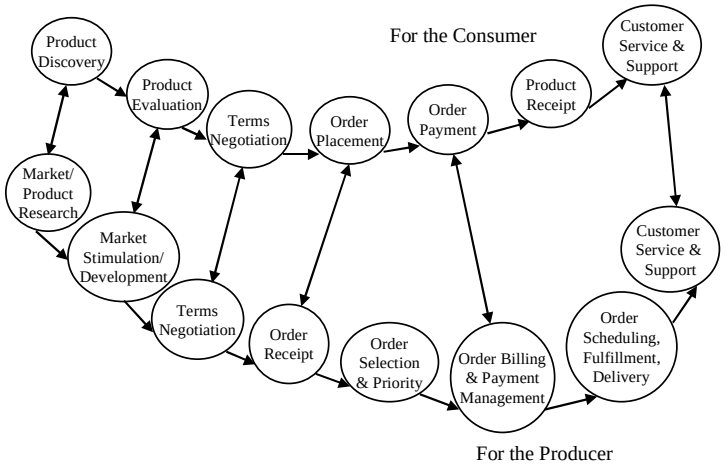

Figure 2. E-commerce Value Chain

The second deficiency is the lack of a comprehensive life cycle model for automated negotiation. Business negotiation is not a once-in-a-lifetime activity. It should be viewed as a continuous, iterative process in which the negotiation outcomes of the current and past negotiations can and should affect the future choice of negotiation policies and strategies and, thus, the behavior of an automated negotiation system. A comprehensive negotiation life cycle model is needed to clearly define the different phases of a

negotiation process and to show 1) what information and knowledge should be specified or defined at different phases, 2) how they can be used by an automated negotiation system to conduct its negotiations with other automated negotiation systems on behalf of its clients, and 3) how the results of negotiations provide the feedback to other phases of the life cycle.

In this report, we present a model of automated e-business negotiation, which identifies the key concepts and elements needed to build an automated negotiation system. One of the elements in the model is a formal decision model, which is missing in many implemented automated negotiation systems (including the EECOMS's negotiation server). The decision model formally defines negotiation contexts, policies, goals, strategies, plans of decision or action and their inter-relationships. A comprehensive model of a negotiation life cycle is also presented. Based on these two models, a system architecture is proposed. It consists of software components that are needed to capture the key concepts of the models and to implement an automated negotiation system for supporting bi-lateral bargaining-type of business negotiations. The new architecture is an extension of the system architecture of the existing EECOMS's negotiation server.

The organization of this report is as follows: In Section 2, some related R&D efforts on negotiations and commercial systems are surveyed. Section 3 presents the model of automated e-business negotiation. Section 4 presents the formal decision model and then focuses on the formal specification of negotiation policies and shows how negotiation contextual specifications are related to negotiation goals. It also presents the formal specification of negotiation strategies and shows how negotiation goals are related to the plans of decisions or actions to be taken by an automated negotiation system. Section 5 presents a comprehensive model of a negotiation life cycle. Section 6 describes the new system architecture. Section 7 outlines an implementation plan. Section 8 concludes the report and outlines some problems for future research.

2. Literature Survey

Negotiation has been a popular topic which has been investigated by people in various disciplines such as economics, social sciences, psychology, game theory, negotiation support systems, agent technology, and machine learning. People negotiate about a large variety of subjects, such as diplomatic issues, international conflicts, family affairs, meeting schedules, production plans, purchase of goods, and the acquisition of services, etc. In this work, we are interested in business negotiations in the e-business environment. In this section, we survey some related work on negotiations.

2.1 Social Sciences

Pruitt [PRU81] studied negotiations from the social-psychological point of view. The book deals with human psychology which is involved in a human-based negotiation. Much attention is paid to the motives, perceptions, and other micro-processes underlying the behavior of a negotiator and to the results of laboratory experiments on negotiation. The strategic choice model, which is related to the negotiation decision model in our work, is presented in the book. The strategic choice model states that a bargainer must choose among three basic strategies to move toward an agreement. The first strategy is to

concede unilaterally, the second strategy is to stand firm and ask the other party to concede, and the third strategy is to collaborate with the other party in search of a mutually acceptable solution. These are very general strategies. An automated negotiation system needs to have more specific strategies on how to concede, how much to concede, and how fast to concede (i.e., the rate of concession). We shall address these issues in our work.

The work reported in Raiffa's book [RAI82] divides negotiations into several categories according to the number of parties and the number of issues involved: two parties/one issue, two parties/many issues, or many parties/many issues. According to Raiffa, different categories of issues 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. Sandholm [SAN96] studied the coalition formation problem in the context of distributed artificial intelligence and multi-agent systems (MAS). Raiffa used case studies to illustrate the link between "negotiation as a science" and "negotiation as an art". Negotiation policy is not explicitly mentioned in the book, but parts of the book discuss negotiation strategies and tactics. For example, the book emphasizes the importance of the preparation for an initial (first) offer, which is one of the dimensions of negotiation strategy space to be discussed in our work.

Several books [LAX86, KAR93, SHE99] address and offer practical negotiation policies and negotiation strategies. However, the distinction between "policy" and "strategy" is not clear. In fact, these terms are often used interchangeably in books. One general advice (or strategy) for a seller is to offer a high price, to make slow concessions during the bargaining stage, and to concede at the end to make the deal. The general advice for a buyer is to offer a low price and to gradually increase the offer. However, the advice is not a panacea. It can backfire in some cases. Since it basically advises the negotiator to set a "high goal" that is difficult to achieve, the negotiator may not be satisfied with the outcome even though the outcome is favorable from an objective point of view. Furthermore, tension and dissatisfaction can build up between negotiation parties if the difference is large and it takes a long time to approach an agreement. The proper strategy to use can depend on several factors. For example, if one of the goals is to establish a long-term relationship with the counterpart, it would be better to make a relatively large concession in order to show goodwill.

There is a related work on the application of negotiation principles in the domain of labor management disputes. Sycara and her colleagues at Carnegie Mellon University developed PERSUADER [SYC85, SYC90], which provides a framework for intelligent computer-supported conflict resolution through negotiation/mediation. The framework integrates AI and decision theoretic techniques to provide enhanced conflict resolution and negotiation support in a group problem-solving setting. PERSUADER, acting as a mediator, facilitates the disputants' problem-solving so that a mutually agreed settlement can be achieved. It embodies a general negotiation model that handles multi-party, multi-issue, single, or repeated encounters based on the integration of case-based reasoning [SYC88a] and multi-attribute utility theory [SYC88b].

2.2 Fixed Pricing, Auction, Reverse Auction and Bargaining

Fixed pricing, auction, and reverse auction are different forms of business negotiations. Fixed pricing is the simplest form of negotiation. In a fixed pricing transaction, after a buyer or a seller posts its price, together with other business terms, either on the Internet or in printed catalogs, a seller or a buyer has only one option: "take-it-or-leave-it". Suppose a seller wants to sell thousands of identical and low-value items, it is obviously not feasible for the seller to bargain with thousands of customers. Therefore fixed pricing is a good choice and is widely used in business-to-customer (B2C) e-commerce. Regular retailers such as Wal-Mart and online retailers such as Amazon.com both use the fixed-pricing scheme to sell products. On the other hand, if a potential business deal involves a large quantity of high-value products, bargaining over the unit price is often a must because a small difference in the unit price can make a big difference in the total cost.

According to McAfee and McMillan [MCA87], "auction is a market institution with an explicit set of rules determining resource allocation and prices on the basis of bids from the market participants". The auctioneer usually starts the auction with an initial offer, then bidders submit their bids in response to the initial offer or bids from other bidders. The auction ends according to some established rules. There are different auction protocols for different situations. The auction can be divided into two types: sealed-bid auction and open auction. Two widely used open auctions are the open-cry English auction in which price goes up and the Dutch auction in which price goes down. AuctionBot [WUR98] from the University of Michigan is a configurable, flexible, and scalable auction server that supports both software and human agents in auctions. [OMA98] describes the design of API specifications for AuctionBot. If the API specification is implemented, developers can develop software agents that can interact with the auction server. Auction theory is a branch of economics of which a detailed discussion is beyond the scope of this report. A good overview on auction theory can be found in [MIL82, MIL89].

Reverse auction is similar to auction, but the auctioneer is a buyer instead of a seller. One typical form of reverse auction is that a buyer issues RFQs (Request for Quotes) to multiple sellers. The CNP (Contract Net Protocol) [SMI80], an early and popular negotiation protocol in distributed AI and multi-agent systems, is essentially a reverse auction for task distribution. There are some commercial systems for reverse auctions. They will be reviewed in the following subsections.

Given the growing popularity of auctions over the Internet, some people claim that auction can replace negotiation on the Internet [SEG98]. While Internet auction provides efficiencies by allowing people from different places to join the auction process, auction usually focuses on one issue: price. It usually involves the sale/purchase of a single item; however, in many business transactions, price is not always the only concern of a buyer--the quality of the product/service, the delivery date, the method of payment, the return policy, the warranty, etc., are all important considerations. Auction systems usually do not allow negotiations over multiple issues. Bargaining is the most complicated form of negotiation. It is the focus of this work. Bargaining involves the

search for a new acceptable alternative and the concession of negotiation parties. When there is a conflict between two negotiation parties, if it is possible to find a new alternative that satisfies both sides, the new alternative is taken. Otherwise, concessions of either one side or both sides are necessary to reach an agreement. One significant difference between bargaining and auction is that, in an auction, only one –side (either buyer or seller) is doing the concession. The other major difference is that multiple issues can be involved in bargaining.

2.3 Game Theory

Game theory [BIN99, JON80] is the mathematical study of conflicts. Since negotiation is used for resolving conflicts, game theory has been applied to analyze negotiation processes. It focuses on the prediction of whether or not an agreement can be reached and if so, the nature of that agreement. Another focus of game theory is the negotiation mechanism design: the definition of protocols that limit the possible strategies used by players and the mechanism to achieve Pareto optimal outcome for all the negotiation participants [ZLO96]. This is useful since, in game theory, it is assumed that the negotiation participants are rational and have complete information about the other players. If game theory were able to derive several (preferably only one) strategies that are suitable for the negotiation, the negotiator would mechanically follow the strategies and get the optimal outcome. Unfortunately, in the case of real world negotiations, rationality and complete information assumptions are usually not valid. The lack of any of the assumptions may lead to a wrong predication in the game theory paradigm. [MYE83] has shown that, in the situation of incomplete information, a rational game player (negotiator) may fail to reach an agreement despite the existence of an agreement zone. Moreover, since game theory is mainly used to predict the outcome of the negotiation process, it can not determine how to obtain the outcome through interactions between players or how to select an optimal negotiation strategy in a negotiation process.

2.4 Negotiation Support System

Recently, people are trying to use computers and networks to support (aid), or even automate the negotiation process. A Negotiation Support System (NSS) [KER99, RAN97] is a kind of computer system that assists human negotiators in carrying out a negotiation process. NSSs are usually based on a phase model of negotiation [KER99]. In the phase model, a negotiation process is divided into three phases: preparation (or pre-negotiation) phase, bargaining phase, and post-settlement phase. In the preparation phase, the system solicits the preferences of the individual users and constructs utilities. The main purpose of the preparation stage is to let the users have a better understanding of his/her real preferences. In the bargaining phase, users exchange structured proposals and/or free style messages. NSSs usually provide an asynchronous communication channel so that both negotiators do not have to be online at the same time. When negotiators have reached an agreement in the bargaining phase, they have an option to enter the post-settlement phase. The post-settlement phase uses a third-party program to check whether the agreement is Pareto-optimal or not. If it is not, the program can suggest possible Pareto-optimal solutions that are more desirable than the agreement

reached by both sides. If there is more than one Pareto-optimal suggestion, and the negotiators have different preferences about these suggestions, they can enter into another round of negotiation.

Lim proposed a theoretical model of NSSs in [LIM93]. The paper divides a NSS into two major components: decision aid component (i.e., DSS) and electronic communications component. Due to the information processing capability of the decision aid component, solutions with NSS is better than those without NSS in terms of the distance from the Nash solution, the distance from the efficient frontier, and the confidence over the final outcome. Compared with a verbal communication channel, an electronic communications channel is more "task-oriented" than "social-oriented". Therefore, the time to settlement is reduced and the satisfaction with the system is higher. Like decision support systems (DSSs), the focus of NSSs is still on "support". There is no facility for negotiators to specify their negotiation policies and strategies. The human negotiators are expected to apply their own negotiation policies and strategies when composing offers and counter offers. The focus on NSS is to support the negotiator in a negotiation process, not to make a decision by computer.

Lo and Kersten proposed an integrated negotiation environment incorporating NSS with software agents in [LO99]. The negotiation support component of the environment is based on INSS (InterNeg Support System, http://interneg.carleton.ca/interneg/tools/inss/). There are two negotiation agents: individual negotiation software agent (INSA) and co-operative software agent (COSA). INSA provides assistance to the individual negotiator. It uses case-based reasoning (CBR) to inform the negotiator of related negotiation cases and possible actions to be taken. It also helps the negotiator to elicit preferences and construct utilities. Furthermore, INSA uses data mining and statistical methods to extract negotiation knowledge from historic negotiation data. COSA acts as a mediator in the environment and provides the user with information on the possible integrative moves.

Benyoucef and Keller propose the concept of Combined Negotiation Support System (CNSS) in [BEN00]. A user may be interested in many goods or services in a typical business deal. Consequently, the user may engage in many negotiations concurrently. Although negotiations seem to be independent of each other, the goods and services are usually interdependent. There is a need to coordinate and reconcile these negotiations. Different negotiations are modeled as different negotiation agents, and the monitoring and control of these agents are done by a workflow system.

2.5 Agent Technology

Barbuceanu and his colleagues at the University of Toronto proposed a generic negotiation shell in [BAR99]. The negotiation shell is constructed to distribute ownership over resources and activities among agents. It contains a process component, a behavior representation component, and a reasoning and decision making component. The process component, which is based on the work reported in [BAR97], deals with the structure of the interactions among agents. The process component is similar to our previous work on negotiation protocols and negotiation primitives [SU00b]. The

behavior representation component allows agents to represent their own goals and behaviors, as well as to model the counterpart's goals and behaviors. Unlike our work on negotiation policy and strategy, the goals in the negotiation shell do not distinguish "high-level" goals and "low-level" goals. The reasoning and decision making component allows agents to deliberate about goals, behaviors and outcomes. It makes use of constraint optimization and relaxation techniques presented in [BEC94].

Kasbah [CHA96] is a Web-based, multi-agent marketplace where users create buying and selling agents to help exchange goods. A user wanting to buy or sell goods can create an agent, initialize it with a particular negotiation strategy, and send it off into the marketplace. Kasbah's agents proactively seek out potential buyers or sellers and negotiate with them on behalf of their owners. Each agent's goal is to complete an acceptable deal, subject to a set of user-specified constraints, such as a desired price, a highest (or lowest) acceptable price, and a date by which to complete the transaction. Negotiation between buying and selling agents in Kasbah is single-issue (i.e., price), and bilateral. After buying and selling agents are matched up, the only valid action for buying agents to take is to offer a bid to the seller. Selling agents respond with either a binding "yes" or "no". Kasbah provides buyers with one of three negotiation strategies: "anxious", "cool-headed", and "frugal"--corresponding to a linear, quadratic, or exponential function, respectively, for increasing a bid for a product over time. Similar negotiation strategies are provided for sellers. The simplicity of these negotiation heuristics makes it intuitive for users to understand how their agents are behaving in the marketplace, but these negotiation heuristics are too simple even in the viewpoint of the developers. As pointed out by the developers, the agent sometimes makes apparently "stupid" decisions.

Sierra, Faratin and Jennings present a formal model of negotiation between autonomous agents in [SIE97]. The model defines a range of strategies and tactics that agents can employ to generate initial offers, to evaluate proposals, and to generate counter proposals. The intelligence of a negotiation agent is manifested by the use of an evaluation function for each negotiation attribute (a simple aggregation function), and a simple hand-coded monotonic concession negotiation strategy. A proof on the convergence (i.e., acceptance condition) of negotiation is presented in the paper.

2.6 Machine Learning

The field of machine learning attempts to let software agents learn how to negotiate by themselves, rather than attempting to exhaustively implement human negotiation strategies in software agents. In [ZEN98], Zeng and Sycara present Bazaar, an experimental system for updating negotiation offers between two intelligent agents during a bilateral negotiation. The paper contains a formal analysis of the negotiation state space, which is capable of tracking a rich set of issues and tradeoffs that are necessary for multi-issue negotiation. It explicitly models negotiation as a sequential decision making task and uses Bayesian probability as the underlying learning mechanism. The authors present an example by using price as the issue of negotiation.

Another machine learning approach is to use genetic algorithms, based on the principle of Darwinian evolution. 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 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 the strategies are, the higher the probabilities that they are chosen as parents. Also, mutations may be randomly introduced. The size of the initial population, the number of generations, the crossover rates, and the mutation rate are parameters of the algorithm. In principle, after many trials, the negotiation agent is able to derive a good strategy; however, the actual result depends on many elements, e.g., the number of training trials, the algorithm parameter configuration, and the quality of the negotiation strategy of the training opponent. For instance, to achieve a good strategy, the number of trials varies from about 20 generations [OLI97] to 4000 generations [DWO96] and all runs must be made against opponent(s) whose strategies are as realistic as possible.

The UMBC group in the EECOMS project proposed a rule learning approach reported in [UMB00]. UMBC uses decision trees to learn negotiation rules. The approach works as follows: At first, a set of training data is gathered. The training data is either artificially generated or obtained with the help of negotiation experts from human-based negotiations. Then, the data is fed into a decision-tree learning package, such as C5.0 from RuleQuest Research (www.rulequest.com). Finally, a set of rules is converted from the decision tree built by C5.0. The major reported obstacle is that it is difficult to get realistic and good training data.

The machine learning approach to negotiation is promising in theory since it is able to derive negotiation strategies based on learning algorithms. Therefore, negotiators are freed from manually designing and implementing good strategies. However, it is not clear how to incorporate the approach used in Bazaar in the multi-issue bargaining-type negotiation system developed in EECOMS, since the example given in Bazaar only deals with a single attribute: price. The genetic algorithm approach may be too slow to train a negotiation system that would be of practical use in real-world negotiations.

2.7 OMG Negotiation Facility

OSM (Open Service Management, www.osm.net) is an active member of the Object Management Group (OMG) and has submitted a proposal to OMG in response to the Electronic Commerce Domain Task Force's (ECDTF) Negotiation Facility Request For Proposal. From OSM's perspective, the life cycle of Electronic Commerce transactions goes through the phases of discovery, convergence, agreement, and engagement. The proposal from OSM addresses the convergence, agreement and engagement phases of this life cycle in the context of declared negotiation policies, rights, and obligations. The proposal defines three collaborative models: bilateral negotiation, multilateral negotiation, and promissory commitment. The first two models are discussed here.

Components of the proposal have been deployed in systems supporting on-line gambling, mediated negotiation, rights management, and auctioning applications. The proposal defines negotiation within the framework of CORBA.

Six states are defined for a bilateral negotiation state transition model. There are three intermediate states, "requested," "proposed," and "offered"; and three terminal states "accepted," "rejected," and "timeout." The difference between "offered" and "proposed" is that "offered" signifies that the content of the received proposal may be agreed to but should not be changed, while "proposed" allows a counter-offer. The state transition model is simpler than our state transition diagram presented in Appendix A because some negotiation situations are not considered. The proposal does not specify the content of the offer. There is no decision-making mechanism for deciding under what conditions the offer should be accepted or rejected. Multilateral negotiation is a voting system where the number of yes/no votes determines the outcome (agreed, rejected, or withdrawn) of a motion presented to the negotiation system.

2.8 Commercial Systems

Priceline (www.priceline.com) is an example of reverse auction. 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. Priceline collects consumer demands for a particular product or service at a price set by the customer and communicates that demand directly to participating sellers or to sellers' private database systems. Consumers agree to hold their offers open for a specified period of time to enable Priceline to fulfill their offers from inventories provided by the participating sellers. Once fulfilled, offers generally cannot be canceled. By requiring consumers to be flexible with respect to brands, sellers and/or product features, Priceline enables sellers to generate incremental revenue without disrupting their existing distribution channels or retail pricing structures.

Traditional Request-For-Quotes (RFQs) or Request-For-Proposals (RFPs) are processed manually. Perfect (www.perfect.com) 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. Immediately after submitting an RFQ, a buyer will receive an ordered list of suppliers, most likely to satisfy the needs specified in the RFQ. If the buyer is dissatisfied with the results, the buyer can adjust his/her search criteria and submit another search. The RFQ supports multiple dimensions, instead of one dimension: price. The RFQ processing engine also supports a multiple-item RFQ.

Perfect has an iterative negotiation mechanism, which allows negotiation parties to pass RFQ attachments (e.g., text messages, diagrams, product requirements, or information about existing equipment) as online documents between them using an electronic communications channel without any decision support. This is different from our

negotiation proposals, which are highly structured messages that are processed and used by negotiation servers for decision-making.

HaggleWare (www.haggleware.com) provides pricing decisions that match buyers and sellers who are engaged in real-time, online negotiations. It uses electronic negotiation characters like Chester or Maria, to act as an electronic sales representative on behalf of the retailer. HaggleWare provides clients with a Web interface to transmit information from the client to the core engine and back. The application resolves negotiation disputes and displays electronic negotiation characters together with a free-style message. 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 an unbalanced negotiation system which focuses on price negotiations. The seller side is automated, but the buyer side is manual. MakeUsAnOffer (www.makeusanoffer.com) is a Website that uses the HaggleWare technology to build a "Real-Time Online Haggling" market.

Win Square (www.winxwin.com) is a software package for helping the negotiator find the right tactics for negotiations. 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. Win Square suggests several tactics for almost every negotiation situation and provides examples of typical usage. After reviewing the other party's likely responses, the user can select the best tactic for his/her own customized negotiation plan. Win Square has four sections: Get Started, Analyze a Negotiation, Solve a Problem, and Get Help. Unlike our work, the focus of Win Square is in the preparation stage of a negotiation.

All the above systems support bi-lateral negotiations. One Accord Technologies (www.peacesummit.com/index.html) has a range of products to support both bi-lateral negotiations and multi-lateral negotiations. As a stand-alone system, One Accord Negotiator can help the users to better understand the issues and users' own preferences. The user can also simulate other parties and develop strategies for negotiations. On the One Accord Network, the negotiator puts the user in secure real-time communication with other negotiators and generates optimal solutions based on the preferences of any number of other parties located anywhere in the world. The Professional version is designed for simultaneous facilitation of any number of cases in either the stand-alone mode or the One Accord Network.

In summary, the existing R&D efforts in negotiation have provided good insights into some important concepts and implementation techniques for building automated negotiation systems. A lot of effort has been in the study of negotiation strategies, which serve as the general guidelines for human-based as well as automated negotiations. Although some negotiation strategies have been implemented in the existing negotiation

systems, little work has been done on the formal representations of negotiation contexts, goals, policies, and their relationship with strategies and the detailed plans of decisions and actions followed by a negotiation system. A formal model of automated e-business negotiation and a comprehensive model for negotiation life cycle are still missing.

3. Model of Automated E-business Negotiation

In this section, we present a general model of automated e-business negotiation, which captures the key concepts and elements involved in automated negotiations. The model is an abstraction of an automated negotiation system. It is intended for serving as the framework for R&D efforts in the area.

We define the model of automated e-business negotiation as a 7-tuple <C, AN, M, PT, RCSM, CBESM, DM> where,

1. C stands for clients. Clients in automated negotiation are people who represent the interests of different business enterprises that participate in negotiations. These people may play different roles and serve different functions in negotiations. For example, the role of the Policy Maker may define the goals and general policies for negotiations in different business contexts. The role of the Negotiation Expert may provide the strategies and plans of decisions and actions to be carried out by negotiation servers which conduct negotiations on the behalf of clients. And, the role of the User may monitor the progress of an automated negotiation system and interact with it to provide the needed information or to resolve problems that can not be resolved by the system.

2. AN stands for automated negotiator. An automated negotiator conducts negotiations on behalf of its clients. A single AN may represent both parties who are involved in negotiations. Or, multiple ANs may represent different parties in negotiations. ANs can be implemented as servers in a client-server architectural environment or agents in an agent architectural environment to conduct automated negotiations on behalf of their clients. The EECOMS's negotiation server is an example of the server-approach to implement an AN.

3. M stands for messages. In order for ANs to do negotiations, a set of well-defined messages need to be defined and exchanged between them. The messages should contain meaningful negotiation primitives to express the intents of the sender as well as data and constraints that are useful for the receiver to react to the messages. In the EECOMS project, eight primitive operations (Call-for-Proposal, Propose Proposal, Accept Proposal, Reject Proposal, Modify Proposal, Withdraw Proposal, Terminate Negotiation, and Acknowledge Message) have been introduced. Messages exchanged between replicated EECOMS's negotiation servers are wrapped in a set of XML business object documents (BODs) which conform to the format and structure proposed by the Open Applications Group (OAG). This set of XML negotiation BODs have been recommended to OAG for its inclusion into the existing set of BODs.

4. PT stands for negotiation protocol. To conduct an automated negotiation, ANs must follow a well-defined protocol, which specifies what alternative actions can be taken at different states of a negotiation process. In the EECOMS project, we have defined a bi-lateral negotiation protocol in the form of a finite state diagram [HUA00, SU00a]. It captures the behaviors of both the supplier and the buyer of products or services.

5. RCSM stands for a requirement and constraint specification model. A data model is needed to specify the requirements and constraints associated with a product/service a client is interested in selling/buying. This information needs to be registered with an AN which the client selects to represent his/her interest. The data model can also be used to specify the contents of negotiation messages so that data carried in the messages (e.g., a proposal) can be compared with the registered requirements and constraints to determine if the data and constraints conflict with those of the client. EECOMS's negotiation server uses an Active Object Model (AOM) for requirement and constraint specifications [LEE00].

6. CBESM stands for a Cost Benefit Evaluation and Selection Model. During a negotiation process, an AN may receive a proposal that contains several combinations of product/service features that are acceptable to its counterpart. The AN needs to have a way to evaluate these alternative combinations, rank them in terms of their costs and benefits, and select the best one for acceptance. When generating a counterproposal, the AN needs also to evaluate and rank available options and select the proper one to form the counterproposal. In EECOMS, we have adopted the CBESM proposed in [SU87] and have implemented several proposed preference scoring and aggregation methods in the EECOMS's negotiation server.

7. DM stands for a decision model. As we have pointed out in the introduction section, a formal decision model is needed to capture a business enterprise's negotiation goals, policies, strategies, plans of decisions and actions, and their inter-relationships. Like most of the implemented negotiation systems, the EECOMS's negotiation server implements several rules of concessions, which are used in the generation of counterproposals when some constraint conflicts have been detected. How these low-level concession rules are derived, and how they are related to an enterprise's high-level negotiation goals, policies, strategies, etc., have not been investigated. For this reason, research on a DM has become our main focus. In the next section, we shall present the results of our investigation.

4. Decision Model

We shall give an overview of the key concepts of the decision model and then go into a more detailed discussion on each key concept.

The decision model is defined as a 6-tuple <W, CX, G, L, P, S> where,

1. W is an enterprise's mini-world, which represents the information, material, financial, personnel, and other resources that the enterprise has access to.
2. CX is a set of negotiation contexts specified by an enterprise.
3. G is a set of negotiation goals of the enterprise.
4. L is a set of plans of decision and actions used by an automated negotiator.
5. P stands for negotiation policy, which maps a negotiation context to a negotiation goal.
6. S stands for negotiation strategy, which maps negotiation goals to plans of decision or actions.

Figure 3 shows the inter-relationship of these key concepts. The figure will be explained progressively in the following subsections.

4.1 Enterprise Mini-world, Negotiation Context, and Negotiation Goal

Every business enterprise operates in a **mini-world** of business, in which the enterprise has access to some of the information, material, financial, and personnel resources that exist in the real world. These resources may be available in-house or in other enterprises but accessible to the enterprise. They represent the internal and external conditions and states that the enterprise's business is in. The mini-world of one enterprise can be expected to be quite different from that of others. The mini-world changes constantly and reflects the dynamic nature of an enterprise's business.

An enterprise usually conducts different types of negotiations with many different counterparts under different negotiation conditions or situations. Information about the counterparts and different types of internal and external conditions or situations is important for defining the specific goals of negotiations. For example, information about what types or sizes of companies it deals with, the credit ratings of these companies, the current market conditions, its own inventory, internal and external business events, or other accessible information in the enterprise's mini-world, are all important for setting the goals of negotiations. Some of the information may be stored in the enterprise's local database and/or application systems. Others may be accessible from remote databases or application systems by, for example, calling the methods of remote objects that encapsulate these databases and application systems. Based on the accessible information, an enterprise can define a set of **negotiation contexts**, each of which can be expressed by a Boolean expression whose truth value can be determined by evaluating the expression against the enterprise's mini-world (see Figure 3). These contextual expressions capture the typical negotiation conditions and situations that the enterprise encounters, for which negotiation goals can be specified.

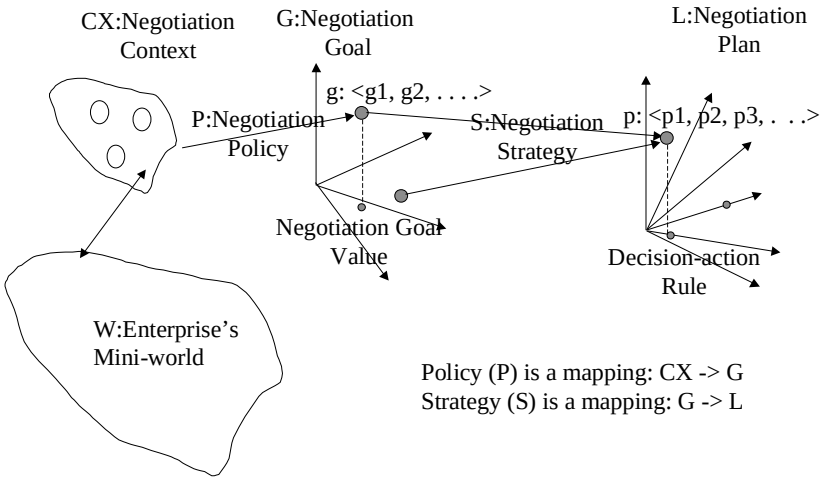

Figure 3. Key Concepts in the Decision Model

The goal that the enterprise wants to achieve in a particular negotiation can be different from one negotiation context to another. For example, the goal of a supplier may be to achieve the maximal profit and to reach an agreement at a short time in a negotiation, if the buyer is a small company, without a good credit, and ordering a small quantity of a product. On the other hand, if the buyer is a very reputable company with good credit and the order is large, the goal may be to take the minimal profit for the purpose of establishing a business relationship with the buyer without any concern over the length of time needed in a negotiation to reach an agreement. The if-conditions in the above examples are negotiation context specifications and negotiation goals are specified in terms of profitability (P), desire-for-relationship (D), and time-to-agreement (T) where P, D, and T are different aspects of a negotiation goal. We shall call these different aspects of a negotiation goal, the **goal dimensions**. An enterprise can define any number of goal dimensions that are deemed necessary to express its goals.

It is desirable to have a quantitative way of specifying negotiation goals with respect to a set of defined goal dimensions. For the above purpose, we first introduce the concept of **goal space**. The goal dimensions defined by an enterprise form a multi-dimensional goal space as shown in Figure 4. Each dimension has an index with a value in the range between 0.0 and 1.0 to serve as a specification of the degree of importance for a company to achieve the negotiation goal in a specific goal dimension. The profitability index is used by the policy maker to specify the importance of one dimension of a business goal, namely, monetary gain. The P value indicates the minimum level of profit that must be made on a deal. It will influence the "bottom line" and the decision-action rules used in a negotiation process. A profitability index P that is close to 1.0 indicates that a high profit must be made before the deal is accepted. If the negotiation party is a supplier, a high

value for P indicates the supplier prefers a high price. If the negotiation party is a buyer, a high P value indicates the buyer prefers a low price. A low P value indicates that, in order to satisfy other goals, the Policy Maker is willing to make a lower profit.

Time is an important factor in most business activities. Thus, time to reach an agreement should be considered as an important dimension of a negotiation goal. In most business negotiations, if the time to reach an agreement is too long or passes a deadline, the final result of the negotiation is no longer relevant. For example, if a company produces a product suitable as Christmas gifts and requires 25 days to manufacture the product, the company may have no interest in any negotiation on raw material purchase if the delivery day is after the first day of December. This goal is represented in the goal space by the time-to-agreement index T. If a quick resolution (either agreement or termination) is desired or required, then the value for T should be specified closer to 0.0. An index T = 1.0 indicates that there is no time limit that the enterprise wants to set for a negotiation.

The desire-for-relationship index represents how strongly the policy makers want to establish a business relationship with the counterpart through this negotiation. An index D = 1.0 means it is a "must have" deal, for whatever the reason. For example, a start-up company is eager to establish a long-term relationship with a Fortune 500 company for future credit reference. Its policy maker would most likely set the D value close or equal to 1.0. A D index value that is close to 0.0 means that a business relationship with the company is not at all important.

A **negotiation goal** is therefore a point in the multi-dimensional goal space, which can be quantitatively represented by a triplet (in our three dimensional example) as (p, t, d), where p, t, and d are values corresponding to the goal dimensions P, T, and D, respectively. It should be noted that P, T, and D are only examples of goal dimensions. Different enterprises can have different types and numbers of goal dimensions.

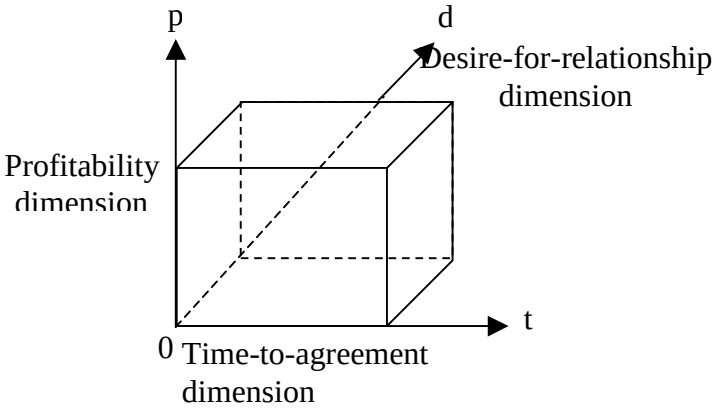

Figure 4. Example of a 3-dimensional Goal Space.

4.2 Negotiation Policy

The term **policy** in English has different meanings in different contexts. In the American Heritage Dictionary of the English Language (Third Edition) [AME96], **policy** has two entries:

- The first entry has three definitions. The first one is "a plan or course of action, as of a government, political party, or business, intended to influence and determine decisions, actions, and other matters." Example policies are American foreign policy and the company's personnel policy. The second one is "a course of action, guiding principle, or procedure considered expedient, prudent, or advantageous." One example is "Honesty is the best policy." The third one is "prudence, shrewdness, or sagacity in practical matters." This definition is often used to modify another noun, such as policy statements and policy issues.

- The second entry has two definitions. The first one is "a written contract or certificate of insurance." An example of such policies is the "return policy" or "refund policy" of purchased goods. This definition of policy is essentially the "terms and conditions" usually specified in purchase contracts. In our opinion, they are a part of the attribute set to be negotiated. For example, "return policy" and "refund policy" can be represented as attributes associated with a negotiation transaction (i.e., transactional attributes instead of product/service attributes). The second one is "a numbers game."

Our definition of **business negotiation policy** is based on the concepts and terms given in the first two definitions of the first entry. A business negotiation policy is:

> "A general guiding principle for achieving a business negotiation goal under some specified conditions. It is intended to influence and determine the decisions or actions to be taken in a business negotiation".

A negotiation policy is a high-level specification of some specific goals that a business enterprise intends to achieve in a negotiation under a specific condition or situation. It is intended to **influence** and **determine** the decisions and actions to be taken in a business negotiation process. It **does not** specify what specific decisions and actions should be taken to achieve the goal. The specifications of decisions and actions for implementing negotiation policies are called negotiation strategies. We shall address negotiation strategies in Section 4.4.

Before we continue our discussion on negotiation policy, we would like to mention some existing works on enterprise policies. Policy specifications have been used in areas such as access control, security management, distributed system and network management. Java (www.javasoft.com) is a modern programming language, which introduces the concepts of **policy** and **permission** to implement an easily-configurable and fine-grained access control mechanism. The Java 2 platform allows Java developers to specify

permissions in a textual policy file. A permission is associated with an access to a system resource. In order for an applet (or an application running with a security manager) to access a resource, the corresponding permission must be explicitly granted to the code that attempts the access. A permission typically has a name (often referred to as a "target name") and, in some cases, a list of one or more actions.

Ponder [DAM00, MAR96], developed at the Imperial College of the University of London, is a high-level language for specifying security and management policies of distributed systems. It can be used to specify security polices with a role-based access control, as well as general-purpose management policies. An application of Ponder for realizing enterprise viewpoint concepts is reported in [LUP00].

A Routing Policy Specification Language (RPSL) [ROU99] is described in an IETF's (Internet Engineering Task Force) RFC (Request For Comments). It allows a network operator to specify routing policies at various levels, such as the Autonomous System (AS) level in the Internet hierarchy. At the same time, policies can be specified in sufficient detail in RPSL so that low-level router configurations can be generated from them. RPSL is extensible, and new routing protocols and new protocol features can be introduced at any time.

The Security Policy Specification Language (SPSL) [SEC00] is an IETF Internet Draft. It is a language designed to express security policies, security domains, and the entities that manage the policies and domains. SPSL currently supports policies for packet filtering, IP Security (IPsec), and Internet Key Exchange (IKE); however, it may easily be extended to express other types of policies.

To the best of our knowledge, there is no existing work on the formal specification of negotiation policy and its relationship with the concepts of negotiation context, goal, strategy, and plan of decision or action, as applied in e-commerce.

Negotiation policies are specifications, which relate specific negotiation contexts to specific goals. The negotiation contexts, goal dimensions, and policies specified by one enterprise can be different from others. It is the responsibility of the people who play the role of **Policy Maker** in an enterprise to define them. We can formally define **Negotiation Policy** as a function P, which maps from the set of negotiation contexts CX to a set of goal points G in the goal space: i.e., P: CX -> G as illustrated in Figure 3. A specific policy can thus be specified by the general expression: If cx_i then g_j where cx_i is a member of CX and g_j is a member of G for some i and j. For example, a general policy defined by a medium-size company can be as follows:

IF (the counterpart is a Fortune 500 company) AND (the counterpart is a company with which this company has previous business relationship) AND (the monetary value of the deal to be negotiated is large)
THEN P = 0.3, T = 0.9, D = 0.9;

The business goal represented by (0.3, 0.9, 0.9) specifies that the company is willing to take less profit for the purpose of establishing a long term relationship with the Fortune 500 company. It is also willing to spend time in a negotiation to reach a deal. The above policy specification is not in a formal language. In our work, we propose to use the rule specification language and GUI tools that have been developed for UF's Event-Trigger-Rule (ETR) server for policy rule specifications.

Figure 3 shows the formal model that captures the relationships among the concepts of negotiation context, policy, goal space, goal, strategy, plan, and decision-action rule. The last three concepts are introduced in the next several subsections.

4.3 Decision-Action Rule

In order to ensure effective and meaningful communication between negotiation servers in an automated negotiation system, the automated negotiators (ANs) must follow a well-defined protocol to exchange negotiation primitives and data during a negotiation process. A negotiation protocol can be conveniently defined by a finite state diagram consisting of a number of states and transitions. An example is given in [HUA00, SU00a], which defines the protocol used for the bi-lateral negotiation used in the present EECOMS's negotiation server. For the reader's convenience, it is given in Appendix A. At each state, an AN needs to make a decision or to take some action based on some conditions before transiting to the next meaningful state. The specification of that conditional decision or action can be in the form of a **decision-action rule**. Generally speaking, there are alternative decision-action rules that can be used by an AN in each transition of a protocol. For example, at the state that a negotiation server receives a proposal from its counterpart (e.g., State S6 in Figure A1 and A2 of Appendix A), a number of alternative decision-action rules can be defined and used for guiding the decision to accept a proposal (e.g., to transit from S6 to S4). Another set of alternative decision-action rules can be defined and used for the decision to reject a proposal (i.e., to transit from S6 to S3). Based on the bi-lateral negotiation protocol given in Appendix A, the transitions in various states of the protocol are shown below:

- The initiation of a negotiation transaction (issuing the initial negotiation proposal or call-for-proposal)
- The acceptance of a proposal
- The rejection of a proposal
- The termination of a negotiation transaction
- The modification of a proposal
- The withdrawal of a proposal
- The generation of a counterproposal

In the following subsections (Sections 4.3.1 to 4.3.7), we shall present some useful alternative decision-action rules associated with the above transitions. Decision-action rules are defined by people in a business enterprise who play the role of the **Negotiation Expert**. They capture the techniques used by skillful human negotiators and are used by an AN to conduct its negotiation on behalf of the enterprise.

4.3.1 Initiation of a Negotiation Transaction

In EECOMS's negotiation protocol, a negotiation transaction instance is started by a negotiation server, which issues either a CFP or a Propose Proposal on behalf of a company. Decisions need to be made with respect to the contents of the CFP or Propose Proposal. Alternative decision-action rules can be used to create the contents. For example, if the negotiation attributes (price, quantity, and delivery date) form part of the data contents of a Propose Proposal, the initial values provided for these attributes can either be very close to what the issuer desires (i.e., the bottom lines) or exceed what the issuer desires with the expectation that the counterpart may bargain them down. Other rules can be applied to determine the proper initial values that are appropriate for achieving a negotiation goal defined in terms of profitability, time-to-agreement, and desire-for-relationship discussed before.

The initiation of a negotiation by a CFP or a Propose Proposal represents two different negotiation styles. If a negotiator is quite familiar with the counterpart and the negotiation environment, the negotiator may start a negotiation by issuing a proposal to the counterpart. On the other hand, if a negotiator is not familiar with the counterpart and the negotiation environment, the negotiator may choose to issue a CFP to ask the counterpart to give the initial offer. The choice of issuing a CFP or a Propose Proposal may depend on the personality of a negotiator. An aggressive and hasty negotiator may choose to issue the initial proposal in order to get the "first move" advantage; however, a cooperative and cautious negotiator may choose not to issue the initial proposal but to wait for the proposal submitted by the counterpart.

4.3.2 Acceptance of a Proposal

Assume that the negotiation server of a buyer receives a Propose Proposal from the negotiation server of a supplier in response to the buyer's CFP or Propose Proposal. After the analysis of its contents, some decision-action rule needs to be applied to guide the acceptance decision. Some examples are given:

a. All attributes of the proposal are within X% difference of the desired values.
b. All major attributes are satisfied.
c. Some attributes are satisfied and some attributes are within X% difference.
d. Global preference score derived from all the attribute values exceeds a certain threshold.

Note that the value assigned to the parameter of a parameterized decision-action rule determines how aggressively a user or organization wants its negotiation server to follow. If X is a small value, the negotiation plan is an aggressive one because the rule requires that the counterpart should move close to the negotiator's position. On the other hand, the plan is a cooperative one if X has a large value. This is because the negotiator is

willing to accept the counterpart's offer that is not close to the negotiator's desired value. Different X values used in the parameterized rules (a and c) represent alternative rules.

4.3.3 Termination of a Negotiation Transaction

The negotiation protocol defined and used in EECOMS's negotiation server provides a set of well-defined "rules of engagement" for both parties to exchange proposals and counterproposals. The acceptance and termination states of the protocol are the states to which a negotiation process should converge. However, the protocol itself does not guarantee that these states will be reached since the negotiation parties may engage in an endless exchange of proposals. To ensure a negotiation transaction's convergence to one of the terminal states, decision rules which explicitly specify the conditions for termination, must be specified and applied. These conditions may depend on the data conditions presented in the previous proposals received. For example, the negotiation server can use the previous exchange of proposals to determine if the negotiation is converging or diverging, and if the counterpart is acting reasonably or not. If not, the negotiation process should be terminated. Some example termination rules are given below:

a. The global preference scores derived for the received proposals indicate that the negotiation is moving away from the desired direction for X number of times.
b. The global preference scores derived for the received proposal are oscillating.
c. The negotiation time exceeds X time units.
d. The number of proposal exchanges exceeds X times.
e. Some combinations of the above.

Decision-action rules for termination provide a quantitative means to make termination decisions, and negotiation history allows a negotiation server to detect the convergence or divergence trend of a negotiation in a traceable and justifiable manner. In order to support such decisions, the architecture of the negotiation server requires a component to record the history of negotiations. The history of an on-going negotiation transaction is required for the enforcement of the above decision-action rules. The history of previous negotiation transactions with a counterpart is useful in the selection of the proper negotiation policy and strategy when negotiating with the counterpart, thus increasing the likelihood that an agreement can be reached in a timely fashion.

4.3.4 Rejection of a Proposal

A negotiation server may decide to reject the current proposal. A rejection is different from a termination in that the negotiation process would continue. It simply informs the supplier's negotiation server that the received proposal is not acceptable, and requests the sender to modify and resubmit a proposal. Some decision-action rules for guiding the rejection decision are given below:

a. The data conditions specified in the proposal are below the acceptable bottom-line values.

b. Insufficient information is provided in the proposal.
c. The data conditions specified in the proposal are acceptable but the receiver of the proposal wants its counterpart to do further concession.

4.3.5 Modification of a Proposal

During the negotiation process, the User who is monitoring the progression of the negotiation may issue a request to his/her negotiation server that he/she wants to modify a proposal that has been sent. The implemented negotiation server supports such a request. Rules can also be defined to guide a negotiation server to make modification to a proposal automatically. For example, if the negotiation server has not received a response for a previous proposal from its counterpart in X time units, a decision-action rule may trigger the modification of the delivery date, the price, etc., specified in the previous proposal. Or, in a company in which prices of products to be sold are updated monthly or quarterly, the negotiator may need to modify the proposal that has been sent if no response has been received and the time is approaching or has reached the month or quarter boundary.

4.3.6 Withdrawal of a Proposal

Similar to a proposal modification, a user may initiate the withdrawal of a proposal. Or rules can be defined to guide a negotiation server in making the withdrawal decision automatically. For example, if the financial condition of the issuer has changed, if the customer has changed his/her order, if the inventory has dropped to a certain level, or if the counterpart has not responded to the previous proposal after X time units, the proposal should be withdrawn.

4.3.7 Counterproposal Generation

Counterproposal generation is one of the most important tasks in a negotiation process. In the analysis of a received proposal, a negotiation server may find that some of the data conditions are not acceptable (meaning, they do not satisfy the requirements and constraints of its client). In that case, some of these conditions need to be changed to form a counterproposal. However, very often, there are many alternative data values that can be used in the counterproposal. A decision needs to be made to select a suitable combination of values to form the counterproposal. In this case, the cost benefit evaluation and selection model must be used to evaluate the alternative combinations and derive the global cost-benefit scores for them. The selection of a proper combination can be guided by a decision-action rule, which can be one of the following:

a. Purely selfish (i.e., select the one with highest global cost-benefit score)
b. Purely unselfish (i.e., select the one with lowest global cost-benefit score)
c. Some point in between.

In generating a counterproposal, a negotiation server may have to do concession based on some decision-action rules. A negotiation position can be lost or gained depending on the concession strategy used. Concessions in negotiation can be broadly classified into two types: static concession and dynamic (or adaptive) concession.

4.3.7.1 Static Concession

Static concession controls the behavior of a negotiation server based on some pre-defined functions. Some examples of static concession are as follows:

One useful function is the Sigmoid function [VON93]: f (x) = 1 / (1 + exp(-g * x)). The sigmoid function results in the following figure (Figure 5) for g = 1.5.

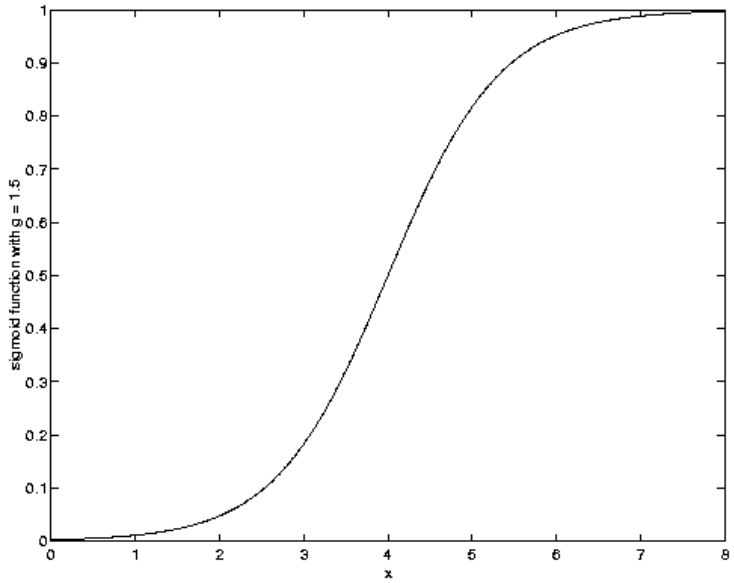

Figure 5. Graph for Sigmoid Function with g = 1.5

As a decision-action rule, this function can be used to control the rate of concession. As shown in the graph, the concession rate is initially very slow. It increases drastically in the middle of the negotiation. Then the concession rate slows down again toward the end of the negotiation when the bottom line value of a negotiated attribute is approaching. By choosing a proper value for g (based on the selected negotiation goal and its mapping to a negotiation plan to be elaborated on Section 4.5), the concession rate can be controlled. By controlling the concession rate, the speed of convergence of a negotiation transaction can also be controlled (e.g., the higher the concession rate, the faster the negotiation can reach an agreement.) The above curve has one turning point (when x = 0). It is possible to define functions that have multiple turning points.

Again, alternative parameter values of a parameterized decision-action rule (e.g., “g” values in this case) determine how aggressively the strategy is to be carried out.

Other functions such as f (x) = k * x, which gives concessions at a constant amount, can be used in a counterproposal generation. Another rule is not to give concession at all (Boulwarism [THO98]) under certain situations. Users can also define arbitrary functions based on user-selected pivotal points and interpolation and extrapolation techniques.

4.3.7.2 Dynamic Concession

In the static approach, the function used for concession is selected and its parameters are set based on a selected decision-action rule. Once selected and set, it will proceed without any regard to what the counterpart does. Dynamic concession strategies take the negotiation behavior of the counterpart into consideration; the function can be adaptive. Two example dynamic concession strategies are explained below.

(a) Assume that the attribute (say, Price) under negotiation is independent of the other attributes. If a negotiation server keeps track of the attribute values presented in the previous proposals sent by its counterpart, it is possible to determine the concession applied by the counterpart (counterpart_concession). The concession of the negotiation server is also known, depending on the static concession function used by the server (own_concession). We can use the following equation for f and use it to control the actual concession:

f = x * own_concession + (1.0 - x) * counterpart_concession

where x is the adjustment factor whose value is in the range of 0.0 to 1.0.

Note that when x = 1.0, counterpart_concession is ignored and the function is reduced to a static concession function. When x = 0.0, own_concession is ignored and f is equal to the counterpart_speed. In the latter case, the server totally adapts the concession speed used by the counterpart. When x is chosen to be some value between 0.0 and 1.0, the concession pattern is a combination of those used by the negotiation server and the counterpart.

There are several ways to compute the counterpart_concession. One simple way is to take the average of the concession values found in the received counterproposals. For example, if the counterpart has the concession value of 20.0, 17.0, 15.0 and 12.0 in the previous four counterproposals, we calculate the counterpart_concession to be (20.0 + 17.0 + 15.0 + 12.0) / 4 = 18.0. Another way is to use extrapolation technique to derive a polynomial curve and get the next value. For example, we can treat the above numbers as four points (20.0, 1), (17.0, 2), (15.0, 3) and (12.0, 4) in the X-Y coordinates, then compute coefficients for a 4-degree polynomial curve. We can then get the Y value for X = 5.

(b) A negotiation server can assign an “importance factor” (from 0.0 to 1.0) to each attribute under negotiation based on the input of its client. For example, 1.0 implies very important and 0.0 implies not important. The server can also compute the “importance

factor" for the same attribute of the counterpart. For example, one heuristic could be that the counterpart's "importance factor" is inversely proportional to the counterpart's concession speed for that attribute. A tactic that can be followed is to compare the "importance factors" associated with all the attributes of both sides. If some of them are different, it would be beneficial to increase the concession speed of those attributes that are not important to the own-side but important to the counterpart-side, and decrease the concession speed of those attributes that are importance to the own-side but not important to the counterpart-side.

In order to support the adaptive concession strategies, the architecture of the negotiation server would need a component to record the history of negotiations. For example, the history of an on-going negotiation transaction (i.e., proposal exchanges) can be used to determine the speed of concession applied by the counterpart (counterpart_concession). In addition, the system can "learn" from experience (i.e., the history of previous negotiation transactions with a counterpart) to generate new policies (i.e., mappings from negotiation contexts to negotiation goals) and strategies (i.e., mappings from negotiation goals to negotiation plans to be explained in the next section) and/or to modify the existing policies and strategies dynamically at run-time.

4.4 Plan of Decision and Action

If D_1, D_2, …D_n (n is equal to 7 in our protocol example) are domains of alternative decision-action rules associated with n transitions, a **negotiation plan** is a combination of n decision-action rules, each of which is selected from a domain of alternative rules for use to guide the decisions and actions of a negotiation server during a negotiation process. Different combinations of decision-action rules form different negotiation plans. Thus, formally, the set of negotiation plans (L) defined by a business organization can be represented by a set of tuples having the general structure $<d_1, d_2, \ldots d_n>$ where d_i in D_i for ($1 <= i <= n$) is a decision-action rule. Conceptually, the transitions of a negotiation protocol form an n dimensional space, which is called the **plan space**. Negotiation plans are points in the plan space as illustrated in Figure 3. Along each dimension or axis, a number of decision-action rules can be defined and arranged in the relative order based on how aggressive or cooperative these rules represent.

4.5 Negotiation Strategy

In order to have a clear and precise definition of "business negotiation strategy," it is useful to first make reference to some definitions of "strategy" as found in the dictionary mentioned in subsection 4.2. They are shown below:

- The science and art of using all the forces of a nation to execute approved plans as effectively as possible during peace or war.
- The science and art of military command as applied to the overall planning and conduct of large-scale combat operations.
- A plan of action resulting from strategy or intended to accomplish a specific goal.

- The art or skill of using stratagems in endeavors such as politics and business.

The third and fourth definitions of "strategy" are closer to what we think the definition of "business negotiation strategy" should be. We define a **business negotiation strategy** as follows:

"A plan of decisions or actions for accomplishing a business negotiation goal."

Having formally defined negotiation goals in Section 4.1 and negotiation plans in Section 4.4, we can formally define a negotiation strategy as a function S: G -> L, where G is a set of goals in the goal space and L is a set of negotiation plans in the plan space, and S maps from G to L. Conceptually, strategies are mappings from some points in the negotiation goal space to some points in the negotiation plan space, as illustrated in Figure 3. Each mapping can be either one-to-one, many-to-one, many-to-many, or one-to-many. In the case of a one-one mapping, a person who plays the role of the **negotiation expert** in a business enterprise has identified a specific goal to achieve and knows a specific plan of decisions or actions, which can be used by a negotiation server to achieve the goal. In the case of a many-one mapping, the expert specifies that a number of goals can be achieved by a negotiation plan. For example, a strategy specification may state that if P dimension in the goal space is in the range of (0.2, 0.5), T in the range of (0.6, 0.7), and D in the range of (0.6, 0.8), then the negotiation plan should use the decision-action rule R1 for the initiation of a negotiation transaction, parameterized rule R5 with parameter value set to 0.05 for the acceptance of a proposal, R11 for the termination of a proposal, etc. The range specifications for P, T, and D in effect allow a number of goals to be mapped to a specific negotiation plan. In the many-to-many mapping case, multiple negotiation plans can be applied to achieve multiple negotiation goals. In the one-to-many mapping case, a goal can be realized by multiple plans. This represents the case that different negotiation experts have different opinions on how to achieve a specific goal.

Although the specification of a negotiation strategy can be one of the above four types, when a business enterprise enters into a particular negotiation, the enterprise's mini-world is defined and the negotiation contexts specified in policy rules can be verified against the mini-world. A specific goal will be selected based on a selected policy, which maps the verified negotiation context to the goal. A strategy specification may map the goal to one or multiple plans. In the latter case, the negotiation server would need the input of a human to select one from the alternative plans because a single plan with a combination of decision-action rules should be used for driving the negotiation process of this negotiation transaction in its initiation, rejection, termination, etc. At run-time, if the enterprise's mini-world has been changed, new policy and strategy may have to be applied to determine a new plan of decisions or actions. A negotiation system and its architecture should support such dynamic changes.

4.6 Section Summary and Related Work

We will now summarize what we have presented in this long section. Referring back to Figure 3, we view negotiation policies as general specifications of different negotiation

goals that a business enterprise aims to achieve under various negotiation contexts (i.e., negotiation policies map contexts to goals). Negotiation policies can be defined by people in the enterprise who play the role of Policy Makers in the form of policy rules. Negotiation strategies can be viewed as specification of specific plans of decisions or actions (to be taken in various steps of a negotiation protocol) that are needed to achieve the goals (i.e., negotiation strategies map goals to plans of decisions or actions). Negotiation strategies can be defined by people who play the role of Negotiation Experts in the form of strategic rules. A powerful rule specification language can be designed and implemented to specify policy rules, strategic rules, as well as decision-action rules. A selected set of decision-action rules, which forms a negotiation plan, can be used to control the behavior of an automated negotiation server in the course of a negotiation.

In a dynamic business environment such as in B2B e-commerce, the enterprise's mini-world can change. Any change of the mini-world may entail changes to the negotiation policy, the strategy, and thus the decision-action rules that should be applied in an on-going negotiation or a future negotiation. The selection of the proper new policy can be accomplished by matching the contextual expressions of policy rules against the contents of the new mini-world. The new policy has a new goal, which in turn determines a new strategy, which in turn determines the new plan of decision-action to be used by an automated system. Some changes in the mini-world may require that new policy and strategy be introduced. Others may require only a change to the policy but not to the strategy or vise versa. The separation of policy-mapping between negotiation contexts and goals and strategy-mapping between goals and plans gives the flexibility in making changes to either one or both.

We believe that it is also important to provide a quantitative way for specifying alternative values in various goal dimensions as well as alternative values in plan dimensions (e.g., by using parameterized decision-action rules). Continuous numerical values instead of an enumeration of some discrete values provide policy and strategy designers a finer granularity of policy and strategy specifications.

To relate what we have presented in this section with some existing works, we note that the term "negotiation strategies or tactics" used in the existing work we surveyed correspond to our decision-action rules. They form a subset of the decision-action rules suggested in this report. For example, negotiation strategies in Kasbah [CHA96] are essentially price decay functions that seller agents use to lower the asking price (or to raise the bidding price for buyer agents) over a given time frame. These strategies fit into the plan dimension of "the generation of a counterproposal" in our decision model. They can be implemented as static concession functions on a single attribute: price. Kasbah does not deal with business negotiation contexts, negotiation goals and high-level negotiation policies. The work presented in [SIE97] discusses negotiation tactics and strategies for service-oriented negotiations among autonomous agents in the ADEPT project. Tactics are functions that determine how to compute the value of quantitative attributes by considering a single criterion, such as time and resource. The paper presents a rich set of well-defined mathematical formula for tactics, which correspond to decision-action rules in our work. It is pointed out in [SIE97] that negotiation strategy is a

function based on the agent's mental states (MSs) and a weight matrix whose rows are attributes to be negotiated and whose columns are different tactics. An agent's MSs contain "information about its beliefs, its knowledge of the environment (time, resources, etc.) and any other attitudes (desires, goals, obligations, intentions, etc)"; however, MSs are not formally defined in the paper. We believe that the so-called MSs correspond to our negotiation contexts, goals, and policies. In our work, we formally define these concepts and their relationships.

5. Negotiation Life Cycle

The concept of life cycle is not new. Life cycles have been introduced for product development and software development. Software development life cycle goes through phases such as requirement analysis, design, implementation, testing, and deployment. Computer-Aided Software Engineering (CASE) tools are programs that aid or automate these phases. Software development in the 1940s and 1950s was like a craftsman's work whose major task was to code using a programming language. As software development became more and more complicated, the craftsman's approach to software development became obsolete. In our opinion, the current status of computer supports for negotiation is very much like the early stage of software development. The main focus has been on the development of a monolithic execution engine capable of generating and exchanging negotiation proposals following a hard-coded negotiation protocol and hard-coded negotiation strategies. The analysis of negotiation contexts, goals, policies, product/service requirements and constraints, and the design of negotiation strategies, plans of decisions or actions, product/service evaluation methods, etc., have not been incorporated in the design and implementation of the existing automated negotiation systems. Also, a post-negotiation analysis to provide feedback to different phases of the negotiation life cycle has not been considered. The state of the art of negotiation life cycle is represented by the two models to be explained below. In our work, we adopt some of the concepts presented in these models and extend them to form our own model of negotiation life cycle.

Three phases are discussed in the life cycle presented in [ROB98], as shown in Figure 6: analysis, interaction design and negotiation implementation. The task of the analysis phase is to describe and formalize the negotiation goals. The task of the interaction design is to plan for achieving the negotiation goals by interactions with the counterparts by using appropriate techniques. The task of the negotiation implementation is to engage in the interactions by using appropriate negotiation protocols and tools. The results from the upstream phases are fed into the downstream phases as the input. The paper discusses the similarity and difference between negotiation life cycle and software life cycle. It also points out that "generally, more automation is provided for the "downstream" stage of negotiation design and implementation". In other words, there is little work on the analysis phase, which roughly corresponds to our work on negotiation policies and goals. The paper discusses negotiations in the context of labor/management negotiations, and no effort was made to implement an automated system to support the negotiation life cycle. The conflict resolution methods discussed in the paper are largely based on alternative

searching, instead of mutual concession, which is frequently used in e-business negotiations.

We extend Robinson's model in two ways. First, we understand that reaching the agreement state or entering the termination state of a negotiation protocol is not the end of a negotiation process. There is a need to analyze the outcome to prepare for future negotiations. A phase called "post-negotiation analysis" is needed after the implementation phase. If an agreement is reached successfully, we need to analyze whether a better deal for both negotiators is possible, and to identify the possible weakness in the negotiation policy and strategy. If a negotiation is terminated unsuccessfully, we need to figure out what factors contributed to the failure. Is it because the price is too high (for sellers) or too low (for buyers)? Is it because the delivery date too soon (for the seller) to meet the demand, or too late (for the buyer) to wait for? Second, some output from the downstream phases should be fed back into the upstream phases. The purpose of the feedback is to influence future negotiations. Most negotiation support systems (NSSs) are constructed based on a phase model of negotiation [KER99]. The phase model divides the negotiation into three phases: pre-negotiation, conduct of negotiation, and (optional) post-settlement, as discussed in Section 2.4.

We combine the phases of the above two models and introduce a four-phased negotiation life cycle model, as shown in Figure 6. The figure also shows how our four phases relate to those of the two models. Our model divides the negotiation process into four phases: analysis, design, execution, and post-negotiation analysis. The analysis phase mainly deals with the specifications of negotiation contexts, policies, and goals by people who play the role of the policy maker. In this phase, the requirements and constraints associated with the products or services that an enterprise purchases or provides are also defined. The design phase deals with the design and specification of alternative decision-action rules to be used by a negotiation system and strategic rules for mapping negotiation goals to plans of decisions or actions. This phase also involves the specification of preference scoring and aggregation methods to be used for cost-benefit analysis, evaluation, and selection of alternative data conditions found in a negotiation proposal. The activities in this phase are to capture relevant rules and evaluation methods to be used by a negotiation system in the next phase. The execution phase deals with the processing of negotiation transactions in an automated negotiation system by following the negotiation protocol, the selected decision-action rules that form a negotiation plan, the requirements and constraints associated with a product or service specified by a business enterprise (the client of a negotiation server), and information obtained from the enterprise for cost-benefit analysis purposes. The outcomes of negotiation processes are gathered and used in the post-negotiation analysis phase to provide feedback to all the preceding phases. Thus, the outcomes (i.e., statistical and historical information of negotiations) may change the policies, strategies, plans of decisions or actions and, therefore, the behavior of the negotiation server in subsequent negotiation processes.

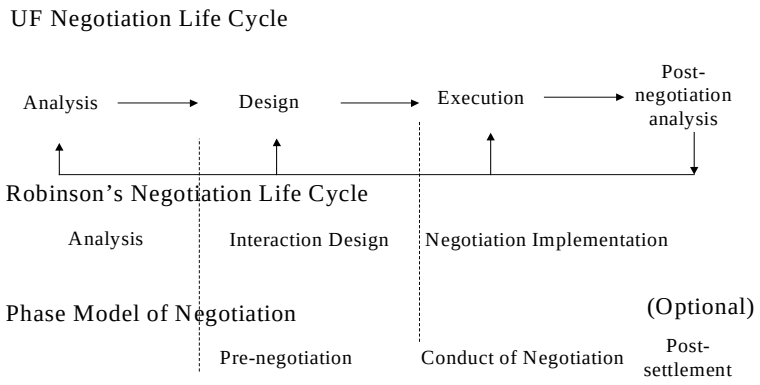

Figure 6. UF's Negotiation Life Cycle and Its Relationship with Two Existing Models

People who play different roles are responsible for different phases of the life cycle model. Three roles have been identified in our model: the Policy Maker establishes policies and goals, the Negotiation Expert designs decision-action rules and strategies, and the User of a negotiation system is responsible for monitoring and interaction with the negotiation system during a negotiation process. These roles can be played by individuals or groups of people. In general, corporate executives would play the role of the Policy Maker. Negotiation practitioners, purchase managers, and sales managers would play the role of the Negotiation Expert. Sales persons and purchasers would play the role of the User. A person may play multiple roles. For example, a corporate executive may be the Policy Maker, the Negotiation Expert, as well as the User.

Figure 6 also shows the approximate correspondence among these models. Our negotiation life cycle model is the only one that includes feedback mechanism. Robinson's negotiation life cycle model does not have a phase after negotiation implementation. Part of the reason is that Robinson's life cycle mainly focuses on each independent negotiation. The Pre-negotiation phase in NSS deals with preference solicitation and utility construction. It covers only part of our analysis and design phases.

6. System Architecture

The system architecture of the current EECOMS's negotiation server needs to be extended to include facilities for specifying and managing negotiation contexts, negotiation goal dimensions, negotiation policies, negotiation plans, decision-action rules and strategies. A new component for managing negotiation history and statistics and for providing a feedback mechanism to other phases of the proposed negotiation life cycle is also needed. Before presenting the new system architecture, an overview of the current EECOMS negotiation server architecture is described so that the readers can understand how the additional components introduced in the new system architecture relate to some of the components in the existing architecture.

6.1 System Architecture of the EECOMS Negotiation Server

The architecture of the rule-based, automated negotiation server developed at the University of Florida under the EECOMS project is described in [HUA00, SU00a, HAM00]. This server can be replicated and installed at multiple sites of the Internet, as illustrated in Figure 7. Two parties conduct a negotiation through the EECOMS communication infrastructure on the Internet, using negotiation primitives and negotiation objects (encapsulated in BODs). One party initiates the negotiation process by submitting an initial product or service request to its negotiation server (e.g., NSA). The negotiation server NSA initiates a negotiation transaction by creating and sending an XML-wrapped call-for-proposal (CFP) or Propose Proposal to the negotiation server (NSB) which represents the other party. Since, in a bargaining process, Propose Proposal is sent back and forth between two negotiation servers, we shall consider the case that NSA issues a Propose Proposal to NSB. NSB uses the data and constraints of the proposal to verify the availability of the goods or services in question. NSB also evaluates the requirements and constraints specified in the proposal against its client's requirements and constraints to determine if there are conflicts in their requirements and constraints. If a conflict is found, a corresponding event is posted to trigger a decision-action rule to resolve the conflict. Currently, the decision-action rule can do one of the following: (1) reject the proposal and suggest possible changes to NSA's proposal, (2) ask a human (a negotiation monitor) to intervene and to make some decision, (3) generate a counterproposal to be returned to NSA, or (4) modify or withdraw the proposal that was sent. If a counterproposal is generated, it is sent back to NSA together with NSB's constraints. This starts another round of bargaining. This process will continue until an agreement is reached or either side unilaterally terminates the negotiation process.

The system architecture of EECOMS's negotiation server and its components are shown in Figure 8. The functions of each component are briefly described below.

(1) ***Negotiation Transaction Manager*** includes:

Negotiation Scheduler : Responsible for initiating a new negotiation transaction / session when the Negotiation Server receives a transaction / session message.

Negotiation Session Processor : Responsible for processing a negotiation session.

A *negotiation transaction* is defined as a sequence of negotiation steps carried out by a pair of negotiation servers that lead to an agreement or disagreement in a negotiation process. Each negotiation step is called a *negotiation session*.

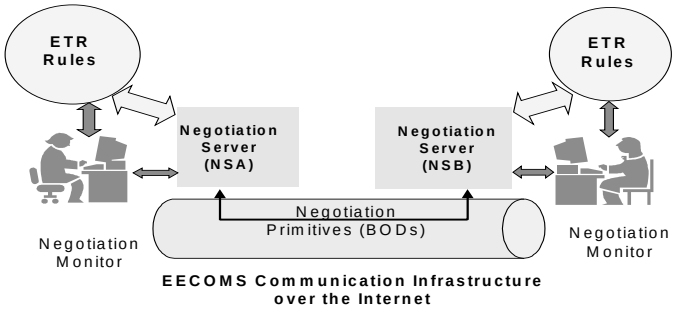

Figure 7. Current EECOMS Negotiation Architecture

(2) ***Constraint Satisfaction Processor (CSP)***: Responsible for evaluating the constraints given in a negotiation proposal or counterproposal against the pre-registered constraints of a negotiation party. It is capable of identifying the constraints that have been violated and posting events to trigger the processing of decision-action rules.

(3) ***Event/Trigger/Rule (ETR) Server***: Responsible for receiving events from the Negotiation Transaction Manager and triggers the proper decision-action rules to relax constraints, to inform the user, etc.

(4) ***Cost/Benefit Module (CBM)***: Responsible for performing cost-benefit evaluation of alternatives based on the pre-registered preference scoring and aggregation methods provided by the negotiation parties.

(5) BOD Converter: Responsible for bi-directionally converting XML BODs and internal message objects used by the Negotiation Server.

(6) ***Negotiation Messaging***: Provides the sender and receiver for Negotiation Servers to communicate with each other using a push communication model.

(7) ***Negotiation Repository***: Provides a persistent storage to store a variety of negotiation data.

(8) Console Manager: Responsible for managing all messages to be notified to Web-based Negotiation Console. It also provides a log management for all negotiation transactions for future use.

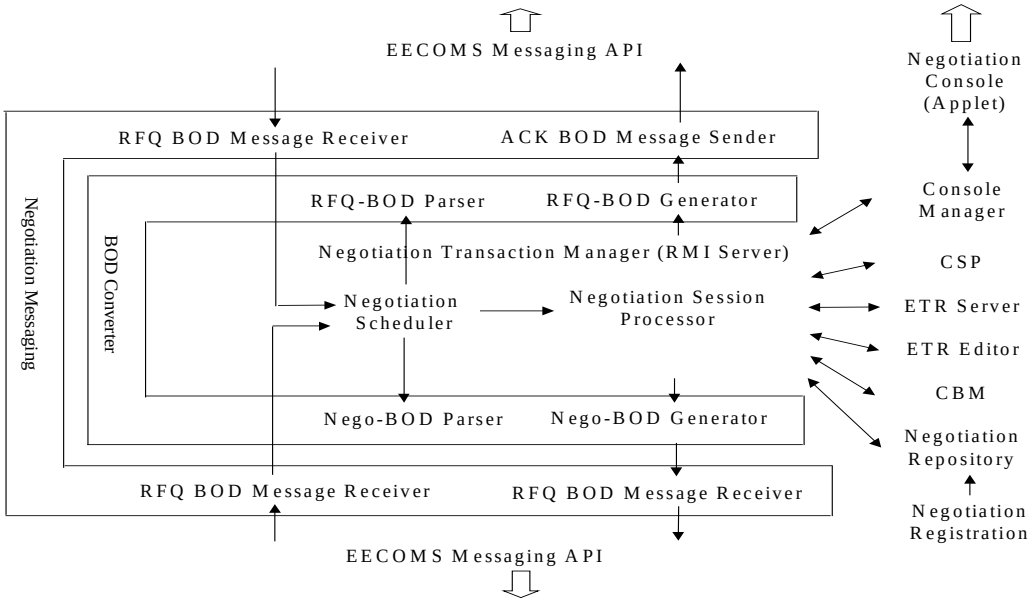

Figure 8. Components of EECOMS Negotiation Server

In addition to the above components, the Negotiation Server has the following utility components:

(1) ***Negotiation Console***: Applet-based program that can be started from a Web browser and is responsible for receiving and displaying negotiation messages, data, rule and XML BOD from the Negotiation Server.

(2) ***Negotiation Registration***: A user interface which is HTML page-based and provides a means for users to pre-register their constraint-based requirements, and information relevant to cost benefit analysis.

(3) ***Event/Trigger/Rule (ETR) Editor***: A web-based tool responsible for defining and editing events, triggers and rules, and generating corresponding Java classes for run-time execution of rules. This component will be used both for event/trigger/rule definition at build-time and for dynamic rule editing at run-time to change negotiation strategy dynamically.

6.2 New System Architecture

Figure 9 shows the new system architecture. The current negotiation server is shown as a dotted box in the new system architecture. Changes to the current negotiation server need to be made in order to deal with new concepts such as policies and strategies. Some of the key components (Transaction Manager, Scheduler and Session Processor) of the existing architecture are shown in the figure. The new components to be implemented are shown in gray color. This figure distinguishes the pre-negotiation activities (arrows

with lower-case letters), negotiation activities (arrows with numbers) and post-negotiation activities (arrows with upper-case letters).

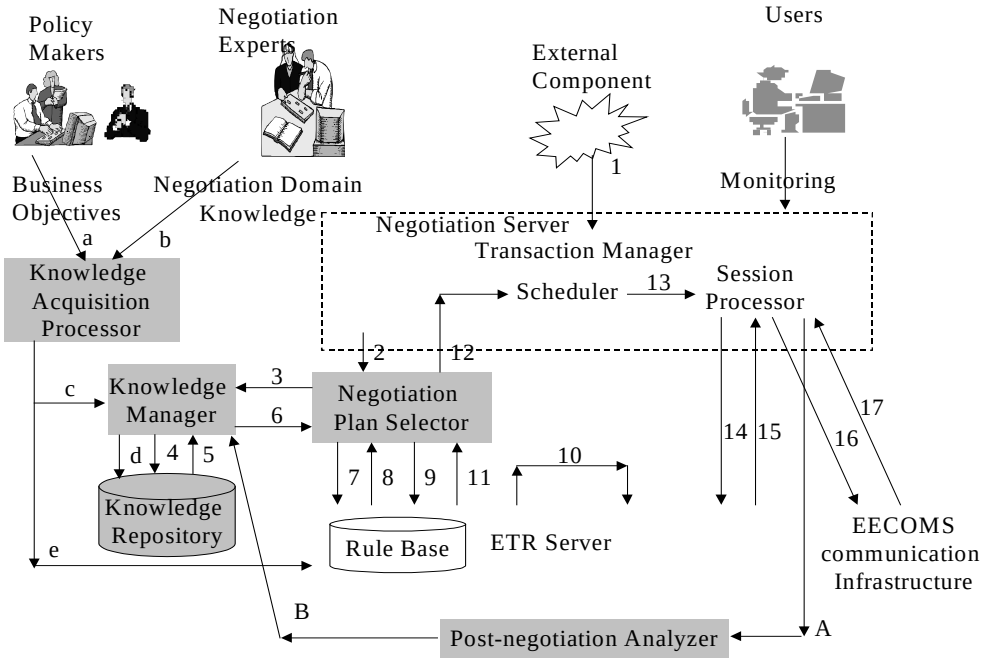

Figure 9. New System Architecture

The three negotiation roles discussed before are shown in the figure. The Policy Maker contributes to the system by defining business objectives (negotiation contextual expressions, policies, and goals) and product/service requirements and constraints. The Negotiation Expert uses domain knowledge and negotiation expertise to define negotiation strategies and plans of decisions or actions, and preference scoring methods and aggregation functions. Policy rules, strategic rules and decision-action rules defined by the Negotiation Expert are stored in the rule base of the ETR server. The User of a negotiation server monitors the negotiation process at run-time, and interacts with the server whenever necessary. The knowledge acquired from the policy maker and the negotiation expert by the knowledge acquisition component is stored in the knowledge repository. The knowledge repository also stores data that constitute the enterprise's mini-world and historical data and statistic information of negotiations.

The Policy Maker and the Negotiation Expert use the GUI facilities provided by the Knowledge Acquisition Processor to input their knowledge before any negotiation transaction begins. These activities are shown by arrows a and b. The acquired knowledge is stored into the Knowledge Repository through the Knowledge Manager (arrows c and d) and the rule base of the ETR server (arrow e). When the Policy Maker

and the Negotiation Expert complete their tasks, the negotiation system is ready to accept negotiation requests.

It is assumed that the negotiation process is requested by an external component such as an ERP system, an agent, or an application system when negotiation services are needed. Arrows with numbers show the negotiation activities. Activities 1 to 13 are the per-transaction activities, and activities 14 to 17 are the per-session activities. Since a negotiation transaction may involve multiple negotiation sessions (a session is a transmission of a negotiation message), activities 14 to 17 may be executed multiple times. However, activities 1 to 13 are executed only once for each negotiation transaction. The Negotiation Transaction Manger receives a request from an external system and starts a negotiation transaction (Activity 1). The request would identify the requesting enterprise and provide some information about what is to be negotiated (e.g., the specification of the product/service that the requester wants to buy/sell). The Negotiation Transaction Manger would then activate the Negotiation Plan Selector (NPS) and ask it to select the proper plan of decisions or actions for this particular negotiation transaction (Activity 2). To do that, the Negotiation Plan Selector has to perform a number of activities. It would make use of the information that came with the initial request to identify what information stored in the Knowledge Repository is relevant to this particular negotiation (Activities 3 to 6). The Negotiation Plan Selector would then post an event (Activity 7) to trigger the evaluation of contextual expressions of policy rules stored in the ETR server's rule base. The contextual expression of a policy rule that evaluates to be true based on the identified relevant information would determine the goal and goal values for this transaction. The negotiation goal and its goal values are returned to the NPS (Activity 8). It is possible that multiple policy rules match with the identified relevant information. In that case, the plan selector would get multiple sets of goal values from the ETR server. The NPS would derive an aggregated set of goal values based on some given criteria (e.g., take the average), post another event (Activity 9) to the ETR server and pass it the final set of goal values. The ETR server would match the goal values against the condition part of strategic rules. The matched strategic rule determines the proper plan of decisions or actions, which is specified by a set of decision-action rules. The ETR server would make these rules active, i.e., ready for use by the negotiation server (Activity 10). After the decision-action rules have been made active, the ETR server would notify the NPS (Activity 11), which in turn notifies the Scheduler (Activity 12) to start the negotiation transaction by following the negotiation protocol.

The Scheduler activates the Session Processor to process the task at each state of the protocol (Activity 13). It would consult with the ETR server for each transition decision (Activity 14) by posting an event to trigger the proper decision-action rule. The ETR server would return the decision result to the Session Processor (Activity 15), which then composes a proper negotiation message based on the decision and sends it through the EECOMS communication infrastructure (Activity 16) to the counterpart's negotiation server. The Session Processor would then wait for the response of the counterpart's negotiation server (Activity 17). Activities 14 to 17 are repeated multiple times in a number of proposal and counterproposal exchanges. Post-negotiation activities are triggered after a negotiation transaction reaches one of its two termination states

(agreement reached or unilateral termination of the negotiation process). The result of a negotiation transaction and its statistic information (e.g., number of message exchanges, the time it takes fore each session, etc.) are stored in the Knowledge Repository through the Knowledge Manager. The information is used in the future selection of policies, strategies and negotiation plans (Activities A and B of the figure).

7. Implementation Plan

A large part of the new system architecture can be implemented by integrating the existing negotiation server with some software components developed at the Database Systems R&D Center (DSRDC) of the University of Florida (UF) and some additional components yet to be implemented.

7.1 Knowledge Acquisition Processor

The GUI tools developed for a Knowledge Profile Manager (KPM) can be used by the Policy Maker and the Negotiation Expert to define negotiation policy rules, strategic rules, and decision-action rules. All these types of rules can be specified in the ETR format and managed by UF's ETR server. Additional facilities are needed to input information such as preference scoring methods, constraints, and aggregation functions. The Registration Manager, a module developed for the previous EECOMS negotiation demos, can be extended for this purpose.

7.2 Knowledge Manager and Knowledge Repository

The Knowledge Manager is a program for managing the information stored in the Knowledge Repository. The Knowledge Repository can be implemented using the serialization facility of the Java language, or the Persistent Object Management (POM) being developed at DSRDC. Data retrieval and manipulation requests issued by the Knowledge Manager to the Knowledge Repository can be posted in an SQL-like query language. The Knowledge Manager provides a set of APIs to the Negotiation Plan Selector, the Knowledge Acquisition Processor, and the Post-negotiation Analyzer to access and manipulate its contents.

7.3 Negotiation Plan Selector

The Negotiation Plan Selector is a new component yet to be implemented. It posts an event to the ETR server when the actual negotiation begins. All policy rules are evaluated when the ETR server receives the event to determine which one of the policy rules' contextual expressions satisfies the current state of the enterprise's mini-world. Recall that the condition part of a policy rule specifies a negotiation contextual expression and the action part of the rule sets the appropriate goal dimension values if the condition part of the policy is evaluated to be true. Ideally only one policy rule is evaluated to be true. However, if multiple policy rules are evaluated to be true, a conflict resolution mechanism (such as taking the average) can be applied to calculate the goal dimension values. After the dimension values are derived from policy rule(s), these

values are evaluated against the conditions specified in strategic rules to determine the proper plan of decision or action. The set of decision-action rules that implement a negotiation plan can then be used to drive the negotiation server.

7.4 Post-negotiation Analyzer

The Post-negotiation Analyzer is a new component yet to be implemented. This component implements the feedback mechanism discussed before. It makes use of the statistic information gathered by the Transaction Manager and updates the knowledge Repository by using the APIs provided by the Knowledge Manager. The contents of the Knowledge Repository represent a part of the enterprise's mini-world. Updating the repository may trigger the processing of the Negotiation Plan Selector to select new policy, strategy, and plan of decision or action for the current transaction.

7.5 ETR Server

We have defined different types of decision-action rules in Section 4. In our implementation plan, the ETR server [LAM98] developed at UF will be used to process these rules. When the negotiation server (see Figure 9) requests the services of the ETR server, it posts events to the ETR server. The ETR server triggers the execution of relevant rules when the server receives the event notifications. The complexity of ETR rules depends on the nature of the rules. Decision-action rules for the acceptance and termination of a negotiation are relatively simple because the conditions for acceptance and termination can be specified by Boolean expressions in the condition part of the rules. The generation of a counter-proposal may involve more complicated computations. The main task in a counter-proposal generation is to calculate concessions for various attributes. Concession calculations may need the support of external method calls. These external method calls can be implemented as Java functions, and they are called in the action or alternative-action part of ETR rules. We expect that decision-action rules for implementing static and dynamic concessions would make an extensive use of external method calls.

8. Conclusion and Future Work

8.1 Conclusion

In this report, we first gave the motivation for conducting a deep investigation into e-business negotiation. We then pointed out that, in spite of many efforts by researchers of various disciplines to investigate different aspects of business negotiations, a formal model of automated negotiation, which captures the key concepts and elements of business negotiation is still lacking. Such a model is important for guiding the future development of automated negotiation systems. In this work, we have defined such a model. One of the key elements in the proposed model is a decision model, which formally defines business enterprises' negotiation contexts, goals, policies, plans of decision or action, strategies, and their inter-relationships. We also proposed a quantitative way to define negotiation goals and, thus, allow more flexibility in mapping

goals to plans of decisions or actions by strategic rule specifications. We also proposed to use parameterized decision-action rules to specify and implement plans of decisions or actions. In this work, we also combined and extended the negotiation life cycle model proposed by Robinson, et al., and the phase model used by several negotiation support systems, and introduced a four-phased life cycle model for automated negotiations. The new model distinguishes the analysis phase, in which information and knowledge such as policies, goals, requirements, and constraints are captured from the Policy Maker -- from the design phase, in which negotiation strategies, plans of decisions-actions, and preference scoring methods and aggregation functions are captured from the Negotiation Expert. The separation of these two phases is important not only because they match better with the different roles that people involved in negotiations play, but also because they provide more flexibility for making changes to contextual expressions and rules that implement policies, strategies, and decision-actions. The design phase provides the specific decision-action rules for the execution phase of an automated negotiation system. The post-negotiation phase provides the needed feedback mechanism to the other three phases and is essential for an automated negotiation system to dynamically adjust and adapt itself to the changing business world.

.

In this report, we have also presented the design of a new system architecture based on the concepts and elements of the automated negotiation model. The components of the architecture and their functionality match with the four phases of the life cycle model. The architecture is an extension of the architecture of the EECOMS's negotiation server. It shows how the additional components for implementing the decision model interact with the implemented components of the existing negotiation server. An implementation plan for the new system architecture has also been outlined in this report.

8.2 Future Work

We have identified two problems that are related to our current effort and warrant further investigation.

8.2.1 Pareto Optimality and Third-party Mediation

Pareto optimality concerns the optimality of a given solution obtained in a negotiation. There can be many possible solutions that can be derived in a bilateral negotiation. The question is whether the particular solution obtained in a negotiation by the negotiation parties is one of the possible Pareto optimal solutions or not. Pareto optimal solutions are solutions that are not "dominated" by other solutions. In the bilateral negotiation situation, two negotiators X and Y bargain over the issues associated with a product or service. A solution is an assignment of values to the issues involved. Generally, there are many possible solutions that have different utilities for the negotiation parties. If we use a two-dimensional diagram to represent the utilities of both parties, the solutions are points in the 2D space. Figure 10 shows the two dimensions; some solutions are shown as points and the Pareto frontier is shown by the curve. The points along an arrow pointing from a solution toward the up-and-right direction are possible solutions that improve the utility of both parties. The Pareto frontier is made up of points or solutions,

which, once reached, can not be further improved (i.e., can not be moved further toward the up-and-right direction). A solution S1 is said to dominate another solution S2 if S1 improves the utilities of both parties, or improves the utility of one party but does not harm that of the other party. It is clear from Figure 10 that C dominates A and D dominates B. E dominates both A and B. However, A does not dominate B and vice versa. C and D are Pareto optimal solutions (points on the Pareto frontier) because there is no way to modify the values of C and D without bringing some disadvantage to one of the parties.

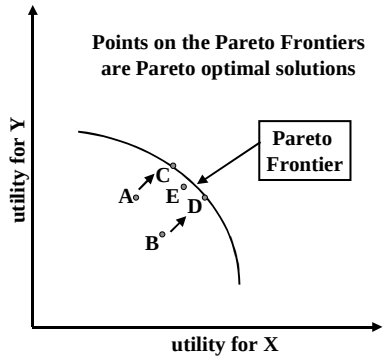

Figure 10. Pareto optimal solutions

Pareto optimal solutions are useful in bilateral negotiations. An agreement (i.e., a solution) reached by the negotiation parties is not necessarily a Pareto optimal solution because both parties may not know the existence of some better solutions. If the requirements, constraints, and preferences of both parties can be made available to a third party (i.e., a mediation software component) or to the post-negotiation component of the proposed system architecture, then the derivation of Pareto optimal solutions based on the solution reached by the negotiation parties can be the function of such a component. In most of the existing works on negotiation support systems (NSSs), a third party mediator is used to find Pareto optimal solutions after the bargaining phase. It is worthwhile to investigate if Pareto optimality can be applied before the bargaining phase to narrow down the number of alternatives that need to be considered by the negotiation parties. For example, if the product/service requirements, constraints, and preferences of both negotiation parties can be made available to a third party, the third party can use this information as well as the information provided by the initial request for negotiation (e.g., a request for quote) to derive Pareto optimal solutions. The negotiation parties can then bargain over the generated alternatives.

The third party mediation approach (either before or after the bargaining phase) works only if both negotiators trust the third party, both are willing to provide the information the third party needs, and both are ready to accept its suggestions. However, there were some experiments and research [CRA99, KOR98] which have shown that some people were not willing to accept the third party solution. There are two possible reasons: The first reason is that negotiators are not willing to share private information with a third

party. The second reason is that negotiators are quite "satisfied" with the *status quo* and there is no incentive to change the solution they have reached.

8.2.2 Multi-bilateral Negotiation

In an e-business negotiation environment, a buyer may negotiate with multiple sellers and a seller may negotiate with multiple buyers concurrently. Bilateral negotiation discussed in this report is just one "thread" in a multi-lateral negotiation effort. In some negotiation situations, it is possible to carry out a multi-bilateral negotiation by multiple bi-lateral negotiations. However, many additional problems need to be dealt with due to the added complexity. The progress and the result of each bi-lateral negotiation need to be coordinated with those of the other bi-lateral negotiations. The decision to be made in one thread of the negotiation may depend on the progress or result of the other. For example, if one supplier is able to supply a large quantity of parts needed at a good price, the buyer may have to terminate some of the on-going negotiations with other suppliers or to reduce the quantity requested from other suppliers. Or, if one or more sellers have decided to make a big concession, the buyer has the power to concede very slowly in negotiations with other sellers. The second example is the idea behind "BATNA" (Best Alternative To the Negotiated Agreement): if it is possible to find a good or better deal with others, why bother bargaining with this "tough negotiator"? Finding more sellers is able to raise the BATNA value of a buyer. Similarly, a seller needs to find more (potential) buyers in order to raise its BATNA value. Compared with bilateral negotiation, multi-bilateral negotiation requires more sophisticated techniques to keep track of the dynamic change of BATNA and to coordinate the concurrent negotiation activities.

References

[AME96] American Heritage Editors, American Heritage Dictionary of the English Language, the Houghton Mifflin Company, Boston, MA, 1996.

[BAR97] M. Barbuceanu and M. S. Fox, "Integrating Communicative Action, Conversations and Decision Theory to Coordinate Agents," Proceedings of Autonomous Agents '97, Marina Del Rey, CA, 1997.

[BAR99] M. Barbuceanu, "A Negotiation Shell," Proceedings of Autonomous Agents '99, Seattle, WA, 1999.

[BEC94] J. C. Beck, "A Schema for Constraint Relaxation with Instantiations for Partial Constraint Satisfaction and Schedule Optimization," Master's Thesis, Department of Computer Science, University of Toronto, 1994.

[BEN00] M. Benyoucef and R. K. Keller, "A Conceptual Architecture for a Combined Negotiation Support System," Proceedings of the Workshop on Negotiations in Electronic Markets, London-Greenwich, Britain, September 2000.

[BIN99] K. Binmore and N. Vulkan, "Applying game theory to automated negotiation," Netnomics, Vol. 1, No., 1, 1999.

[CHA96] A. Chavez and P. Maes, "Kasbah: An Agent Marketplace for Buying and Selling Goods," Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, London, UK, April 1996.

[CRA99] D. Cray and G. E. Kersten, "Negotiating Inefficient Compromises: Is Less Better than More?" 5th International Conference of the Decision Science Institute, Athens, Greece, July 4-7, 1999.

[DAM00] N. Damianou, N. Dulay, E. Lupu, and M. Sloman, "Ponder: A Language for Specifying Security and Management Policies for Distributed Systems", V 2.0 Imperial College Research Report Doc 2000/1, March 2000.

[DWO96] G. Dworman, S. Kimbrough, and J. Laing, "On Automated Discovery of Models Using Genetic Programming in Game-Theoretic Contexts," Journal of Management Information Systems, vol. 12 (3), Winter 1996.

[FEL98] S. Feldman, Keynote speech at ACM 1999 Conference on OOPLSA, November 5, 1999,available at http://www.ibm.com/iac/oopsla99-sifkeynote.pdf

[HAM00] J. Hammer, C. Huang, Y. H. Huang, C. Pluempitiwiriyawej, M. S. Lee, H. Li, L. Wang, Y. Liu, and S. Y. W. Su, "The IDEAL Approach to Internet-based Negotiation for E-Commerce," Proceedings of the International Conference on Data Engineering, San Diego, CA, Feb. 28 – March 3, 2000.

[HUA00] C. Huang, "A Web-based Negotiation Server for Supporting Electronic Commerce," Ph.D. Dissertation, University of Florida, 2000.

[JON80] A. Jones, Game Theory: Mathematical Models of Conflict, Ellis Horwood Ltd., Chichester, England, 1980.

[KAR93] C. L. Karrass, Give and Take: The Complete Guide to Negotiating Strategies and Tactics, HarperCollins Publishers, New York, NY, 1993.

[KER99] G. E. Kersten and S. J. Noronha, "WWW-based Negotiation Support: Design, Implementation, and Use," Decision Support Systems, Vol. 25, No. 2, 1999.

[KOR98] P. Koronen, J. Philips, J. Teich, and J. Wallenius, "Are Pareto Improvements Always Preferred by Negotiators?" Journal of Multi-criteria Decision Analysis, Vol. 7, 1998, pp. 1-2.

[LAM98] H. Lam and S. Y. W. Su, "Component Interoperability in a Virtual Enterprise Using Events/Triggers/Rules," Proceedings of OOSPLA '98 Workshop on Objects, Components, and Virtual Enterprise, Vancouver, BC, Canada, Oct. 18-22, 1998, pp. 47-53.

[LAX86] D. A. Lax and J. K. Sebenius, The Manager as Negotiator: Bargaining for Cooperation and Competitive Gain, The Free Press, New York, NY, 1986.

[LEE00] M. Lee, "Event and Rule Services for Achieving a Web-based Knowledge Network," Ph.D. Dissertation, Computer and Information Science and Engineering, University of Florida, 2000.

[LIM93] L. Lim and I. Benbasat, "A Theoretical Perspective of Negotiation Support Systems," Journal of Management Information Systems, Vol. 9, No. 3, 1993, pp. 27-44.

[LO99] G. Lo and G. E. Kersten, "Negotiation in Electronic Commerce: Integrating Negotiation Support and Software Agent Technologies," 5th Annual Canadian Operational Research Society Conference, Halifax, Nova Scotia, Canada, 1999.

[LUP00] E. Lupu, M. Sloman, N. Dulay, and N. Damianou, "Ponder: Realizing Enterprise Viewpoint Concepts," Proceedings of 4th International Enterprise Distributed Object Computing (EDOC2000), Mukahari, Japan, Sept. 2000.

[MAR96] D. A. Marriotand and M. S. Sloman, "Implementation of a Management Agent for Integrating Obligation Policy," IFIP/IEEE Distributed Systems Operations and Management (DSOM '96), L'Aquila, Italy, October 1996.

[MAE99] P. Maes, R. H. Guttman, and A. G. Moukas, “Agents that Buy and Sell: Transforming Commerce as We Know It,” Communications of the ACM, Vol. 42, No. 3, March 1999, pp. 81-91.

[MCA87] R. P. McAfee and J. McMillan, “Auction and Bidding,” Journal of Economic Literature, 25:699-738, 1987.

[MIL82] P. Milgrom and R. J. Weber, “A Theory of Auctions and Competitive Bidding”, Econometrica, Vol. 50, No. 5, Sept. 1982, pp. 1089-1122.

[MIL89] P. Milgrom, “Auctions and Bidding: A Primer,” Journal of Economic Perspectives, Vol. 3, No. 3, Summer 1989, pp. 3-22.

[MUL96] H. J. Muller, “Negotiation Principles”, Chapter 7, Foundations of Distributed Artificial Intelligence, G. M. P. O’Hare and N. R. Jennings, eds., John Wiley & Sons, Inc, 1996.

[MYE83] R. Myerson and M. Satterthwaite, “Efficient Mechanisms for Bilateral Trading,” Journal of Economic Theory, Vol. 29, 1983, pp. 265-81.

[NIS97] M. Nissen, “The Commerce Model for Electronic Redesign,” Journal of Internet Purchasing, available at http://web.nps.navy.mil/~menissen/papers/jipcomm.htm (July 1997).

[OLI97] J. R. Oliver, “A Machine-Learning Approach to Automated Negotiation and Prospects for Electronic Commerce,” Journal of Management Information Systems, Vol. 13, No. 3, 1997.

[OMA98] K. O'Malley and T. Kelly, “An API for Internet Auctions,” Dr. Dobb's Journal, pp. 70-74, September 1998.

[PRU81] D. G. Pruitt, Negotiation Behavior, Academic Press, New York, NY, 1981.

[RAI82] H. Raiffa, The Art and Science of Negotiation, The Belknap Press of Harvard University Press, Cambridge, MA, 1982.

[RAN97] A. Rangaswamy and G. R. Shell, “Using Computers to Realize Joint Gains in Negotiations: Toward an “Electronic Bargaining Table,” Management Science, Vol. 43, No. 8, 1997, pp. 1147-1163.

[ROB98] W. N. Robinson and V. Volkov, “Supporting the Negotiation Life Cycle,” Communications of the ACM, Vol. 41, No. 5, 1998, pp. 95-102.

[ROU99] Routing Policy Specification Language, available at http://www.ietf.org/rfc/rfc2622.txt?number=2622, 1999.

[SAN96] T. Sandholm, “Negotiation Among Self-interested Computationally Limited Agents,” Ph.D. Dissertation, Department of Computer Science, University of Massachusetts at Amherst, 1996.

[SEC00] Security Policy Specification Language, available at http://www.ietf.org/internet-drafts/draft-ietf-ipsp-spsl-00.txt, 2000.

[SEG98] A. Segev and C. Beam, New Market-based Negotiation Paradigm, available at http://www.haas.berkeley.edu/citm/auction/newnego.html, 1998.

[SHE99] G. R. Shell, “Opening and Making Concessions,” Chapter 9, Bargaining for Advantage: Negotiation Strategies for Reasonable People, Viking, New York, NY, 1999, pp. 156-176.

[SIE97] C. Sierra, P. Faratin, and N. Jennings, “A Service-Oriented Negotiation Model between Autonomous Agents,” Proceedings of 8th European Workshop on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW-97), Ronneby, Sweden, 1997, pp. 17-35.

[SMI80] R. G. Smith, “The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver,” IEEE Transactions on Computers, Vol. C-29, No. 12, 1980, pp. 1104-1113.

[SU00a] S. Y. W. Su, C. Huang, and J. Hammer, “A Replicable Web-based Negotiation Server for E-commerce,” Thirty-third Hawaii International Conference on Systems Science (HICSS-33), Wailea, Maui, Hawaii, Jan. 4-7, 2000.

[SU00b] S. Y. W. Su, “EECOMS’s Negotiation Primitives and Negotiation Protocol,” EECOMS internal document, 2000.

[SU87] S. Y. W. Su, J. Dujmovic, D. S. Batory, S. B. Navathe, and R. Elnicki, “A Cost-Benefit Decision Model: Analysis, Comparison, and Selection of Database Management Systems,” ACM Transactions on Database Systems, Vol. 12, No. 3, Sept. 1987, pp. 472-520.

[SYC85] K. Sycara, “Arguments of Persuasion in Labour Mediation,” Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85), vol. 1, Los Angeles, CA, 1985, pp. 294-296.

[SYC88a] K. Sycara, “Using Case-Based Reasoning for Plan Adaptation and Repair,” Proceedings of the 1988 Case-Based Reasoning Workshop, Clearwater, FL, 1988, pp. 425-434.

[SYC88b] K. Sycara, “Utility Theory in Conflict Resolution,” Annals of Operations Research, Vol. 12, 1988, pp. 65-84.

[SYC90] K. Sycara, “Persuasive Argumentation in Negotiation,” Theory and Decision, Vol. 28, No. 3, 1990, pp. 203-242.

[THO98] L. Thompson, The Mind and Heart of the Negotiator, Prentice Hall, Upper Saddle River, NJ, 1998.

[UMB00] UMBC group, “Negotiation Rules Learning and Related Issue,” EECOMS internal document, 2000.

[VON93] D. Von Seggern, CRC Standard Curves and Surfaces, Boca Raton, FL, CRC Press, 1993.

[WUR98] P. Wurman, M. P. Wellman, and W. E. Walsh, “The Michigan Internet AuctionBot: A Configurable Auction Server for Human Software Agents,” Proceedings of the Second International Conference on Autonomous Agents, Minneapolis, MN, May 1998.

[ZEN98] D. Zeng and K. Sycara, “Bayesian Leaning in Negotiation,” International Journal of Human Computer Systems, Vol. 48, 1998, pp. 125-141.

[ZLO96] G. Zlotkin, S. Jeffrey, and A. Rosenschein, “Mechanism Design for Automated Negotiation and its Application to Task Oriented Domains,” Artificial Intelligence, Vol. 86, 1996, pp. 195-244.

Appendix A: Negotiation Protocol for UF's Negotiation Server

Shown in Table A1 is a list of negotiation primitives used in our Negotiation Server. It overlaps with the list of negotiation primitives given in [MUL96], which includes several additional primitives that are useful for agent communication but are not germane to this work. It also overlaps with the negotiation primitives proposed by FIPA (www.fipa.org) on the primitives CFP, Propose, Reject, and Accept.

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

Table A1: Negotiation Primitives.

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 appendix, we use two state transition diagrams to describe our bilateral bargaining protocol. They are shown in Figures A1 and A2. Figure A1 defines the states and state transitions of the buyer and Figure A2 defines those of the supplier. Several negotiation primitives shown in Table A1 are not included in these diagrams because their usage is self-explanatory.

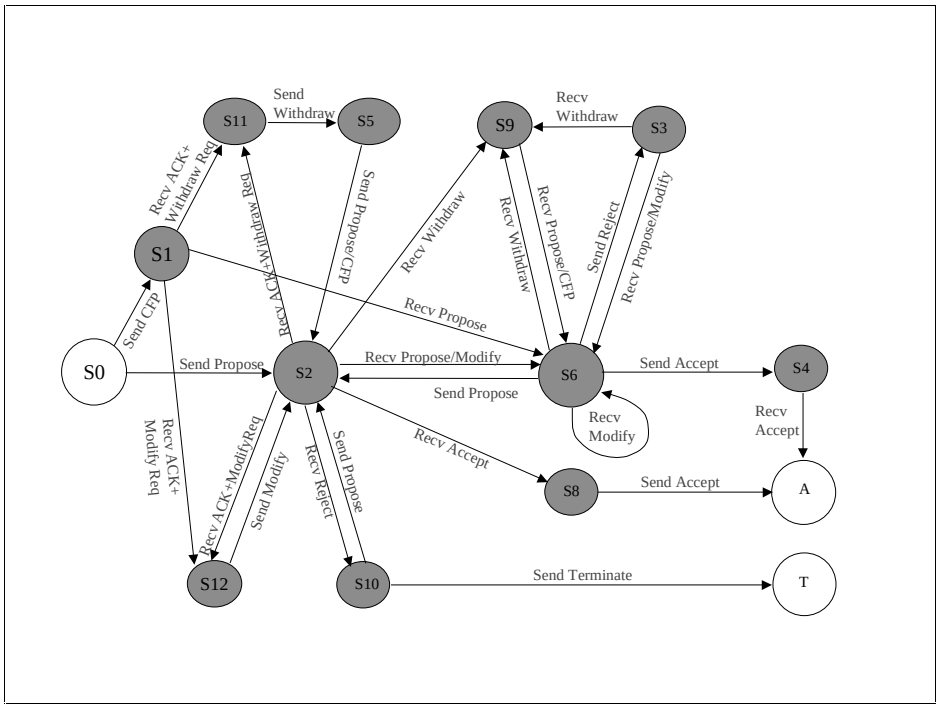

Figure A1. Buyer's Negotiation Protocol State Transition Diagram

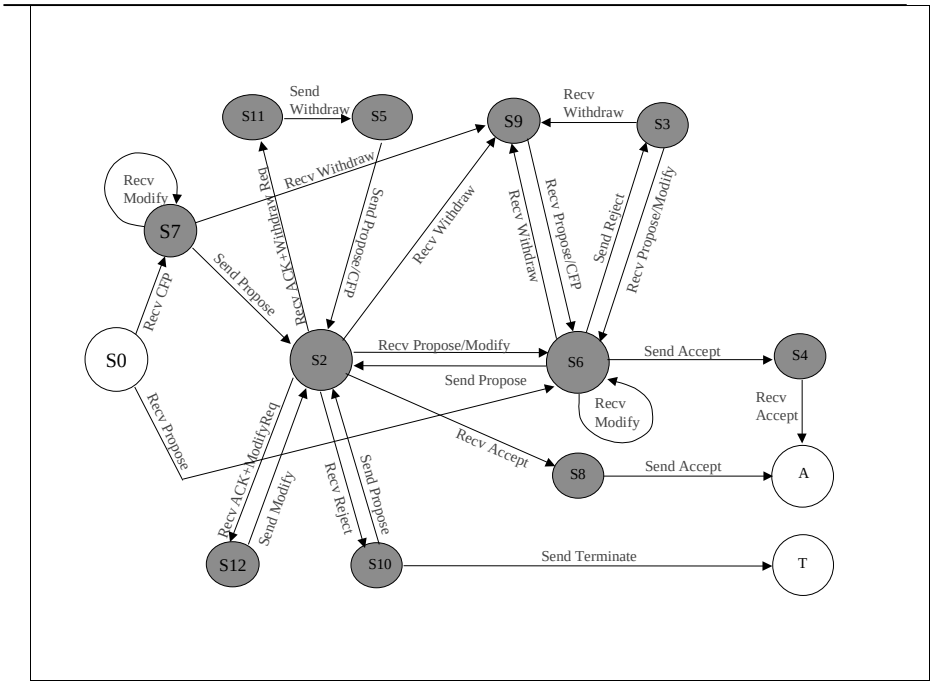

Figure A2. Supplier's Negotiation Protocol State Transition Diagram

There are 15 different negotiation states in which a negotiation server can be 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 when playing both roles. We shall use "buyer" and "supplier" to represent these two roles. The meanings of these states are shown in Table A2.

S0	Initial State
S1	CFP is sent
S2	Propose is sent
S3	Reject is sent
S4	Accept is sent
S5	Withdraw is sent
S6	Propose/CFP/Modify is received
S7	CFP/Modify is received
S8	Accept is received
S9	Withdraw is received
S10	Reject is received
S11	Received the Acknowledgement of the Propose/CFP that was sent; request the Withdrawal of that Propose/CFP
S12	Received the Acknowledgement of the Propose/CFP that was sent; request the Modification of that Propose/CFP
A	Agreement is reached
T	Negotiation is terminated unilaterally

Table A2. Negotiation State Semantics

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 provide some constraints for the negotiation item(s). To respond 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 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. A Reject message is different from a Terminate message. It is used to convey the dissatisfaction of some conditions to the counterpart and expect it 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.