AUTOMATED E-BUSINESS NEGOTIATION: MODEL, LIFE CYCLE, AND SYSTEM ARCHITECTURE

By

HAIFEI LI

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2001

Dedicated to my grandmother Chuntao Lu; my parents, Moshu Li and Fuxiu Liu; my parents-in-law, Junshan Liu and Aimin Feng; my wife, Xiaoli Liu; and my daughter, Joy Li.

ACKNOWLEDGMENTS

I am deeply indebted to my advisor, Dr. Stanley Y. W. Su, who has continuously supported my master's and doctorate studies and research for five years. His academic insight greatly improved the quality of this dissertation.

I thank Dr. Herman Lam for his valuable instruction and assistance. I was a student in two database courses he taught, and greatly benefited from his excellent teaching skills.

I thank Dr. Joachim Hammer, Dr. Abdelsalam Helal, and Dr. Zuo-Jun Shen for being on my supervisory committee. I thank Dr. Sherman Bai and Dr. Michael Frank for their help. Their support and encouragement are gratefully appreciated.

I thank Sharon Grant for her excellent services to the students and faculty members of the Database Systems Research and Development Center and for making it a wonderful place to work.

I thank Dr. Yihua Huang, a visiting professor from Nanjing University, for fruitful discussions, Dr. Chunbo Huang for his early work on the negotiation server, and Youzhong Liu and Jie Meng for their collaborative work.

My special thanks also go to my relatives and friends in China.

This research was supported by IBM at Atlanta, which is the main contractor of a consortium-oriented project funded by the National Institute of Standards and Technology.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

AUTOMATED E-BUSINESS NEGOTIATION: MODEL, LIFE CYCLE, AND SYSTEM ARCHITECTURE

By

Haifei Li

December 2001

Chairman: Dr. Stanley Y. W. Su
Cochairman: Dr. Herman Lam
Major Department: Computer and Information Science and Engineering

Due to the inherent complexity of business negotiations, the automation of e-business negotiation presents a real challenge. Although there is a large body of literature on the topic of negotiation, most studies deal with human-to-human negotiations and a comprehensive model for automated e-business negotiation is still lacking. Furthermore, the existing work in this area does not consider the negotiation process from a full life cycle perspective; therefore valuable information generated by various stages of the life cycle is not fully utilized. Some recent works have attempted to build fully automated negotiation systems and negotiation support systems; however, none of them has demonstrated how decisions and actions taken by the implemented systems to control the automated negotiation process are related to negotiation goals, policies, and strategies.

This dissertation deals with three important issues related to automated e-business negotiations: negotiation model, negotiation life cycle, and system architecture. We formally define automated e-business negotiation as a 6-tuple, which contains clients, automated negotiators, negotiation messages, negotiation protocols, a requirement and constraint specification model, and a negotiation decision model. This dissertation focuses on the negotiation messages and the negotiation decision model. The other elements have been dealt with in a previous work that involves the implementation of an Internet-based negotiation server. Messages passed between replicated negotiation servers are defined in terms of business object documents in the XML format. They correspond to the primitive operations needed to implement the negotiation protocol. The negotiation decision model formally defines the relationship between a business enterprise's mini-world, negotiation contexts, negotiation goals, plans of decisions and actions, negotiation policies, negotiation strategies, and cost-benefit evaluation and selection.

The negotiation life cycle divides the entire negotiation process into four phases: analysis, design, execution, and post-negotiation analysis. The results from the upstream phases are used as inputs in 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 also included. A system architecture based on the negotiation model and the negotiation life cycle model is proposed, and its implementation is described.

CHAPTER 1
INTRODUCTION

Negotiation is traditionally viewed as a playing field for social sciences. In a stereotype of negotiation, two or more negotiators, usually wearing formal business suits, bargain face-to-face for topics of high stakes. The topics vary from corporate purchases, corporate mergers and acquisitions, and labor/management disputes to international relations. Emotional outbursts, bluffs, and ultimatums are considered norms. The negotiation outcomes largely depend on negotiators' negotiation skills. Negotiation skills usually include "soft" skills, such as communication, collaboration, persuasion, personality, aspiration, and ambition. It seems that information technology does not have a place in the area of negotiation.

It is safe to say that some types of negotiations are almost exclusively reserved for human negotiators. Peace negotiation is such an example in which the outcome of the negotiation depends heavily on the perception, aspiration, and diplomatic skills of the political leaders involved in the negotiation. The use of information technology is not so important for peace negotiations; however, some other types of negotiations, such as e-business negotiations, which involve corporate buying and selling over the Internet, are relatively easier to automate. A business enterprise frequently conducts negotiations with suppliers and customers. In business negotiations, negotiation issues (e.g., price, delivery schedule, quantity, and quality) are generally fixed for a particular type of business. It is therefore possible to formalize and automate the negotiation process. According to Morgan Stanley Dean Witter [MOR00], the dollar amount of online business-to-business

(B2B) purchases in 2000 was $200 billion. Morgan Stanley Dean Witter predicted that the amount would grow to $720 billion and $1.4 trillion by 2001 and 2002, respectively. Even if a small percentage of online B2B purchases could take advantage of automated negotiation systems, the potential benefit to the e-business industry could be tremendous.

1.1 Motivation and Scope

In the business world, business deals are often made through some kind of negotiations. In years past, negotiations were human-based. They were carried out by representatives of businesses through face-to-face discussions or by using mail, email, FAX, telephone, or a combination of these. Human-based negotiations can be very costly, lengthy, and error-prone. It would be beneficial if a part or all of the negotiation task could be carried out automatically over the Internet as a part of e-business activities. When necessary, humans can be called upon to resolve problems that cannot be resolved by an automated system.

Automation of business activities is no longer new. Recently, 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 orders electronically. These packages enable companies to reduce processing cost, processing error rate, and processing time to fulfill orders. We believe that electronic negotiation is another essential business activity needed to support the next generation of e-business.

Some negotiations may be 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 solely with a sales manager from IBM. 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. It is the bilateral negotiation that is the focus of our research. We concentrate on automated bilateral negotiations in which structured messages, instead of free text messages, are exchanged between two 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 the bargaining type of negotiation is very general. It can be applied to conduct all sorts of negotiations for products, services, plans, schedules, and the like, 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 such things as price, quantity, delivery date, and payment method and schedule associated with the purchase of a product, or the time, quality, and payment method and schedule of a service.

1.2 Negotiation in E-business

Automated negotiation has become an important topic in the e-commerce community. The terms e-commerce and e-business are used interchangeably in our discussion. Maes et al. [MAE99] of the MIT Media Lab put negotiation at the center of the consumer buying behavior (CBB) model for e-commerce. 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 from whom to buy, (4) negotiation, (5) purchase and delivery, and (6) product service and evaluation.

Nissen [NIS97] of the Naval Postgraduate School made a similar observation about the importance of negotiation in commerce. Nissen proposed the commerce model shown in Figure 1.1. Although he does not use the term “negotiation” in his model, it is apparent that “arrange terms” in the model is roughly 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 an electronic communications channel, the above model is equally applicable to e-commerce.

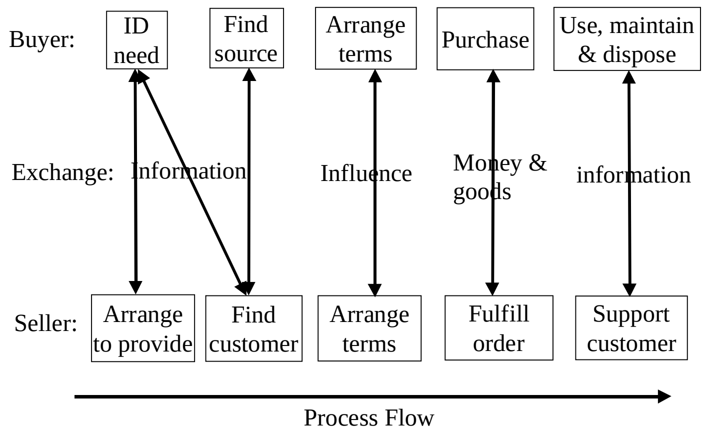

Figure 1.1 Nissen’s Commerce Model

Feldman, Director of the IBM’s Institute for Advanced Commerce, also treats negotiation as an important link in the e-commerce value chain [FEL99]. The value-chain model is shown in Figure 1.2. “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.

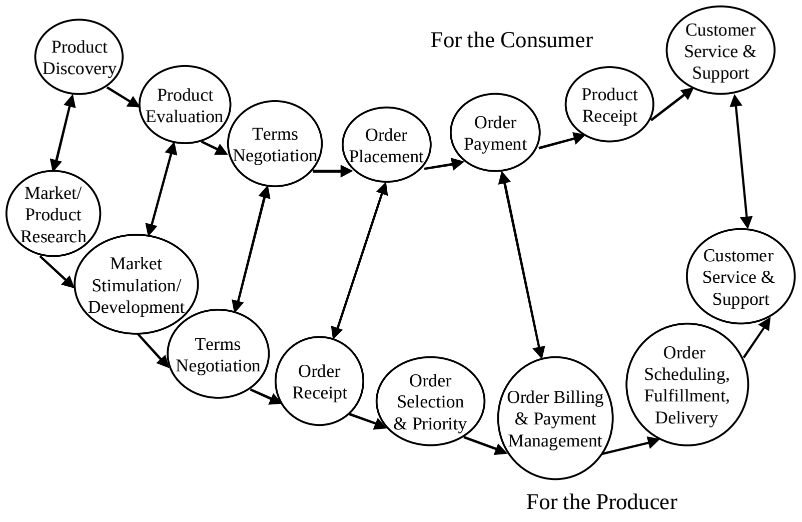
For the Consumer
Product Discovery
Product Evaluation
Terms Negotiation
Order Placement
Order Payment
Product Receipt
Customer Service & Support
Market/ Product Research
Market Stimulation/ Development
Terms Negotiation
Order Receipt
Order Selection & Priority
Order Billing & Payment Management
Order Scheduling, Fulfillment, Delivery
Customer Service & Support
For the Producer

Figure 1.2 E-commerce Value Chain

1.3 Deficiencies in Existing Research and Development Efforts in E-business Negotiation

Based on our investigation of the existing academic projects and commercial systems (to be surveyed in Chapter 2), we have identified two main deficiencies in the current research and development efforts in 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. The formal model, to be presented in Chapter 5, consists of six elements. This dissertation focuses on the two most important elements: the negotiation messages and the decision model. Negotiation messages are a set of structured messages exchanged between negotiators. Based on our literature survey, we found that the set of negotiation messages used in the existing work does not contain all the primitives needed for some negotiation scenarios. For example, the messages do not contain the specification of constraints associated with the product or service being negotiated.

A need exists for a decision model that can capture the relationship among business conditions, negotiation goals, policies, strategies, and decision-action rules that drive an automated negotiation system. Most implemented systems, including the negotiation server implemented at the Database Systems Research and Development Center (DSRDC) at the University of Florida, implement some decisions-action rules (also called in some literature strategies or tactics) without showing how these low-level rules relate to high-level business goals and policies. Decision-action rules are tactic rules that govern how negotiation systems (or human negotiators) behave under certain circumstances. Some examples are the rules for determining the rejection of a negotiation proposal, the amount of concession, and the acceptance of a proposal. They are defined by people who play the role of the Negotiation Expert. Negotiation policies are general business rules defined by people who play the role of the Policy Maker whose purpose is to achieve specific business goals. Negotiation policies reflect the Policy Maker's insight into 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 goals and policies for negotiating with a Fortune 500 company that are different from those set for negotiating with a small business. 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 the 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. What is needed is a formal decision model to capture and relate these concepts.

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 negotiation life cycle.

In this dissertation, we present a model of automated e-business negotiation that identifies the key concepts and elements needed to build an automated negotiation system. One important element of the negotiation model is the specification of a set of negotiation messages to be exchanged between two automated negotiation systems. Another important element of the negotiation model is a formal decision model, which is lacking in many implemented automated negotiation systems, including UF's own negotiation server. The decision model formally defines negotiation contexts, policies, goals, strategies, plans of decision and action, and their inter-relationships. A comprehensive model of a negotiation life cycle is also presented. Based on the negotiation model and the life cycle model, a new system architecture is proposed and implemented. 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 the bargaining type of bilateral business negotiations. The new architecture is an extension of the system architecture of the existing negotiation server.

The organization of this dissertation is as follows: some related research and development efforts in negotiations and commercial systems are surveyed in Chapter 2. Chapter 3 gives an overview of a bilateral negotiation and a formal model of automated e-business negotiation. Chapter 4 presents a set of negotiation primitives and protocols in the form of state transition diagrams for supporting the bargaining type of negotiations. Chapter 5 presents the formal decision model. It focuses on the specifications of negotiation policies as mappings between negotiation contexts and negotiation goals. It also presents the formal specifications of negotiation strategies as mappings between negotiation goals and plans of decisions and actions to be taken by an automated negotiation system. Chapter 6 shows a comprehensive model of a negotiation life cycle. Chapter 7 describes the new system architecture. Chapter 8 outlines the implementation and evaluation of our model based on interviews with people knowledgeable in business negotiations. Chapter 9 some presents conclusions, summarizes the contributions of this research, and suggests some additional avenues for future research.

CHAPTER 2
LITERATURE SURVEY

Negotiation is a popular topic. It has been investigated by people in multiple disciplines, such as economics, social sciences, psychology, game theory, negotiation support systems, agent technology, and machine learning. People negotiate over a large variety of subjects, including diplomatic issues, international conflicts, family affairs, meeting schedules, production plans, purchases of goods, and acquisitions of services. In this research, we are interested in business negotiations in the e-business environment. Here we survey some related work on negotiations.

2.1 Social Sciences

Pruitt [PRU81] studied negotiations from the social-psychological point of view. His study dealt with the psychology involved in human-based negotiation. Much attention was paid to the motives, perceptions, and other micro-processes underlying the behavior of a negotiator, and to the results of laboratory experiments pertinent to negotiations. The strategic choice model, which is related to the negotiation decision model in our work, was presented in his 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. 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 to deal with how to concede, how much to concede, and how quickly 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 a multiple-party negotiation. Research in this dissertation falls into the category of "two parties/many issues." 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 preparing for an initial (i.e., first) offer, which is one of the dimensions of negotiation plans to be discussed in our work.

Several books [LAX86, KAR93, SHE99] address and offer practical negotiation policies and negotiation strategies. However, none makes the distinction between "policy" and "strategy." In fact, these terms were often used interchangeably. One general piece of advice (or strategy) for a seller is to start with a high price, make slow concessions during the bargaining phase, and concede at the end to make the deal. One general piece of advice (or strategy) for a buyer is to start with a low price and to gradually increase the offer. However, this advice is not universally applicable. It can backfire in some cases. Since it generally advises the negotiator to set a "high goal" that may be 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 big and it takes a long time to reach an agreement. The proper strategy to use depends on several factors. For example, if one of the goals is to establish a long-term relationship with the counterpart, it would be better for a negotiator to make a relatively large concession in order to show his/her 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], a system that provides a framework for intelligent computer-supported conflict resolution through negotiations/mediations. The framework integrates artificial intelligence (AI) and decision theory 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 the 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 low-value items. It is obviously not feasible for the seller to bargain with thousands of customers. Therefore, fixed pricing is a good choice for the business-to-customer (B2C) e-commerce. Conventional 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], an 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, and 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] developed at the University of Michigan is a configurable, flexible, and scalable auction server that supports both software and human agents in auctions. O'Malley and Kelly [OMA98] described the design of application programming interface (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 dissertation. A good overview on auction theory can be found in the papers by Milgrom and Weber [MIL82] and Milgrom [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 request for quotes (RFQs) to multiple sellers. The contract net protocol (CNP) [SMI80], an early and popular negotiation protocol in distributed AI and MAS, is essentially a reverse auction system for task distribution. There are already some commercial systems that offer reverse auctions. They will be reviewed in the following sections.

Given the growing popularity of auctions over the Internet, some people claim that auction can replace negotiation on the Internet [SEG98]. While Internet auctions provide efficiencies by allowing people from different places to join the auction process, these auctions usually focus on one issue: price. They usually involve 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, and other aspects are all important considerations. Auction systems usually do not allow negotiations over multiple issues. Bargaining, the most complicated form of negotiation, is the focus of this work. Bargaining involves the search for a new and acceptable alternative and the concession by 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 by either 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, while in a bargaining, both sides can offer concessions. The other major difference between auction and bargaining 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. Game theory focuses primarily on predicting 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 negotiators [ZLO96]. This is useful since, in game theory, it is assumed that the negotiators 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, the assumption on rationality and complete information is usually not valid. The lack of a valid assumption may lead to a wrong predication in the game theory paradigm. Myerson and Satterthwaite [MYE83] have 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 cannot 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 Systems

Recently, researchers have attempted 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 to assist human negotiators in carrying out a negotiation process. It is 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 this phase is to let the users have a better understanding of their real preferences. In the bargaining phase, users exchange structured proposals, free-style messages, or both. Negotiation support systems 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 mediation 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 are at least two Pareto-optimal suggestions and the negotiators have different preferences about these suggestions, they can enter into another round of negotiation.

Lim and Benbasat [LIM93] proposed a theoretical model of NSSs. The paper divides an 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 are 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 counteroffers. The focus on NSS is to support the negotiators in a negotiation process, not to make decisions on behalf of them by computers.

Lo and Kersten [LO99] proposed an integrated negotiation environment incorporating NSS with software agents. The negotiation support component of the environment is based on the InterNeg support system (INSS, available from http://interneg.carleton.ca/interneg/tools/inss). There are two negotiation agents in their work: 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 historical negotiation data. COSA acts as a mediator in the environment and provides the user with information on the possible integrative moves.

Benyoucef and Keller [BEN00] proposed the concept of Combined Negotiation Support System (CNSS). In a typical business deal, a user may be interested in many products or services. Consequently, the user may engage in many negotiations concurrently. Although negotiations seem to be independent of each other, the products and services are usually interdependent. Therefore, 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 [BAR99] proposed a generic negotiation shell. 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 Barbuceanu and Fox [BAR97], deals with the structure of the interactions among agents. Our previous work on negotiation protocols and negotiation primitives [SU00b] is similar to the process component. 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 from "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 by Beck [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," or "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 entirely too simple, even from the viewpoint of the developers. As pointed out by the developers of Kasbah, the agent sometimes made apparently "stupid" decisions.

Sierra et al. [SIE97] presented a formal model of negotiation between autonomous agents. This model defines a range of strategies and tactics that agents can employ to generate initial offers, to evaluate proposals, and to generate counterproposals. 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 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 knowledge in software agents. Zeng and Sycara [ZEN98] presented 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 that is capable of tracking a rich set of issues and tradeoffs that are necessary for a 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, the higher the probabilities that they are chosen as parents. 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.

University of Maryland, Baltimore County (UMBC) [UNI00] proposed a rule learning approach. 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 are either artificially generated or obtained with the help of negotiation experts from human-based negotiations. Then, the data are 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 obstacle reported is the difficulty in obtaining 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 into the multi-issue bargaining type negotiation system, 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 Languages for Negotiation

Agent communication language (ACL) from the Foundation for Intelligent Physical Agents (FIPA, www.fipa.org) has proposed a few primitives for agent negotiations. However, these primitives are not adequate for conducting business negotiations in which back-and-forth bargaining is often needed. For example, FIPA's ACL does not contain primitives for modification and withdrawal of a negotiation proposal, which, more often than not, requires human intervention.

The coordination language (COOL) [BAR95] proposed by Barbuceanu and Fox allows the user to define agents and to manage communication and structured interactions between them. Like FIPA's ACL, COOL is a general language for agent communications. It is not designed specifically for addressing business negotiation problems, and does not provide facilities to aid the negotiation decision-making process.

Muller [MÜL96] published a survey paper on negotiation principles in the context of distributed artificial intelligence (DAI) and MAS. The paper lists the names of several negotiation primitives; however, neither the syntax nor semantics of these primitives is described. Furthermore, concrete negotiation protocol(s) or negotiation scenario(s) are not provided to show the use of these primitives.

As described in Section 2.4, NSSs, such as Negotiation Assistant [RAN97] and INSPIRE [KER99], use a relatively simple language to represent negotiation messages. Both systems use attribute/value pairs to represent proposals (or offers). One example of an offer in INSPIRE is "price = $3.47, delivery = 20 days, payment = 30 days after delivery, returns = 75% refunds with 10% spoilage." NSSs do not provide the language facility to express complex attribute and inter-attribute constraints. Proposals in NSS are used purely by human negotiators to make decisions. As will be shown in subsequent chapters, in our work, negotiation proposals contained in negotiation messages can have complex structures and are for the purpose of both human reading and computer processing.

2.8 Negotiation Standards

2.8.1 OMG Negotiation Facility

Open Service Management (OSM, www.osm.net), a company that is an active member of the Object Management Group (OMG), has submitted a specification 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 specification from OSM addresses the convergence, agreement, and engagement phases of this life cycle in the context of declared negotiation policies, rights, and obligations. The specification 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 specification defines negotiation within the framework of common object request broker architecture (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 offer may be agreed to but should not be changed, while proposed allows a counteroffer. The state transition model is simpler than our state transition diagram to be presented in Chapter 4 because some negotiation situations are not considered. The specification does not define the content of the offer. Content definition is the responsibility of applications using on the specification. 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.

2.8.2 ICE Negotiation Model

The Information and Content Exchange (ICE) standard (www.icestandard.org) is an initiative by IDEAlliance, a non-profit organization supporting the development of industry standards for structured information. The purpose of the ICE standard is to reduce the cost of doing business online and increase the value of B2B relationships. ICE facilitates the controlled exchange and management of electronic assets between networked partners and affiliates. Applications based on ICE allow companies to easily construct syndicated publishing networks, Web superstores, and online reseller channels by establishing website-to-website information networks.

The negotiation model permits a syndicator (i.e., content provider) or subscriber (i.e., content consumer) to define and negotiate parameters of importance to both the subscription and the syndicator/subscriber relationship. A subscriber can choose to either negotiate or not to negotiate with a syndicator. In the non-negotiation case, a subscriber uses the “ice-get-catalog” request to get a catalog, take one of the ice-offers from the catalog, prioritize the parameters, and send it back to the syndicator in a request. In the negotiation case, the subscriber creates an ice-offer in an implementation-dependent way. After the subscriber sends the offer to the syndicator, the syndicator can respond in one of four ways:

1. A code of 200 (OK) and an ice-subscription element. This indicates that the subscriber’s offer is accepted and the subscriber is now subscribed as described in the response.

2. A specific error code: 441 (Counterproposal) and an updated ice-offer. This indicates the syndicator is engaging in parameter negotiation.

3. A specific error code: 440 (Sorry). This indicates the syndicator rejects the proposed subscription offer, but wishes to continue engaging in parameter negotiation.

4. Any other error code. The offer is rejected for the given reason.

The simplest successful negotiation case for establishing a subscription is for the subscriber to issue an "ice-offer" request and the syndicator to respond with a 200 (OK) code. If the syndicator responds with a negotiation response (code 440 or 441), the subscriber enters into a parameter negotiation. As a result, the subscriber can also respond in one of the four ways described above. The back-and-forth negotiation process continues until either an offer (possibly revised) is accepted by both sides, or an offer is rejected by either side.

In an offer from a syndicator to a subscriber or from a subscriber to a syndicator, parameters that are negotiable are labeled as "ice-negotiable." The negotiation may involve multiple parameters, such as subscription operation parameters, subscription business terms, subscription semantic extensions, and protocol semantic extensions. Although the ICE negotiation model is not a general one, it helps us to understand the application of negotiation to specific domains.

2.9 Commercial Systems

This section surveys commercial systems related to business negotiation. Systems' functionalities vary significantly due to the nature of the systems. Some systems are built solely for conducting or supporting online negotiations, while others include negotiation as one of the key components.

LiveExchange online negotiation solutions offered by Moai (www.moai.com) provide the technology and expertise to support auctions, reverse auctions, and structured negotiations. They augment the offline negotiation process by offering tools to assess bids and negotiations based on many factors, such as the quality of goods and the delivery dates. Two important features of LiveExchange structured negotiation are:

1. Multiple Parameter Negotiations: Negotiations can incorporate non-monetary terms, such as shipping time, payment terms, or condition of goods delivered. Those terms can be factored into the overall decision-making process for any business; and

2. Multiple Stage Negotiations: Negotiations can be configured to offer multiple stages of bidding, in which each stage allows for the negotiation of distinct parameters. This functionality allows systems to closely match the offline negotiation process and offers various opportunities at each stage of the negotiation.

The Menerva MarketProcess solutions provided by the Menerva Technologies (www.menerva.com) are a suite of Web-based applications to "enable the whole deal." The solutions extend beyond the scope of ordinary auction-based exchanges. The solutions try to automate the existing business processes, including discovery, negotiation, and transaction/settlement. The solutions have many components and negotiation is one of them. Important features of the Menerva solutions are:

1. Quick adoption and incorporation of business rules, procedures, and processes;

2. Support for public or invitation-only postings;

3. Structured and parameterized negotiation. Buyers and sellers interact not only on price and quantity, but also on all the terms essential to online negotiations, such as product specifications, finance, logistics, and many other parameters;

4. Automatic generation and archiving of business documents: contracts, POs, letters of credit, etc.;

5. XML-based integration with catalogs, decision support systems, payment systems, ERP and other applications; and

6. Repository of all the contracts and deal terms for future reference.

The VerticalNet Structured Negotiation is provided by the VerticalNet Solutions (http://www.verticalnetsolutions.com/solutions/ourproducts/structurednegotiations.asp) Online buyers and sellers can do business and increase their chances of closing transactions with the VerticalNet Structured Negotiation. If a buyer discovers a seller with the right product at the right price, but with an inadequate delivery time, the seller can increase his/her chance of closing the deal by allowing for a negotiation on the delivery time. It becomes a win-win situation.

The VerticalNet Structured Negotiation allows buyers and sellers to negotiate terms in either a structured or unstructured context. This component can be offered as a stand-alone mechanism or as a function of another mechanism, such as VerticalNet Exchange. Key VerticalNet Structured Negotiation features include:

1. *Flexible Taxonomy-Driven Parameters* to enable negotiation parties to locate ideal partners and allow those partners to negotiate on user-defined requirements (e.g., product quality, ship dates, origination points, etc.) versus just cost.

2. *Multiple Concurrent One-on-One Negotiations* to enable an original negotiation party to identify the most favorable terms available through confidential simultaneous negotiations with multiple negotiation parties.

3. *Partial Quantity Acceptance* to enable a negotiation party to split offerings and thereby close deals with at least two negotiation counterparts at a time.

4. *Private/Public Functionality* to allow a negotiation party to specify acceptable negotiation counterparts.

5. *Tracking* logs all negotiation interactions for future reference.

Priceline (www.priceline.com) is an example of a 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 those demands 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. In September 2000, the stock price of Priceline dropped dramatically. Part of the reason was that participating sellers realized that they could do the same tasks by themselves without any difficulty. For example, one of the major services provided by Priceline is to match empty seats in flights and empty rooms in hotels with potential air travelers. However, when airlines and hotels formed their own website (i.e., www.hotwire.com) to sell tickets and hotel rooms, Priceline's revenue from the matching service was reduced significantly.

To help the negotiator find the right tactics for negotiations, Win Square (www.winxwin.com) offers a software package that 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. The focus of Win Square is the preparation phase of a negotiation.

In summary, the existing research and development efforts in negotiation have provided good insights into some important concepts and implementation techniques for building automated negotiation systems. Much effort has been invested in the study of negotiation strategies, which serve as 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, or policies, and their relationships with strategies and detailed plans of decisions and actions followed by a negotiation system. A formal model of automated e-business negotiation and a comprehensive model for a negotiation life cycle are still missing.

CHAPTER 3
AUTOMATED E-BUSINESS NEGOTIATION: OVERVIEW AND MODEL

We present an overview of automated e-business negotiation in order to give readers the background information about the problems addressed in this dissertation. Although an informal description of automated e-business negotiation is useful for understanding the problems to be addressed, a formal model is needed to guide the design and implementation of an automated negotiation system.

3.1 An Overview of Automated E-business Negotiation

3.1.1 Human-based Negotiation Process

Automated business negotiation is intended to solve the same kinds of problems as human-based negotiations. Therefore, it is helpful to briefly review the activities involved in the human-based negotiation process. The description is limited to bilateral negotiation, since this is the most widely used form of business negotiation. In the following discussion, we assume that two negotiators, NA and NB, representing two companies, CA (the buyer) and CB (the supplier), negotiate the purchase of a product or service. The discussion is from the perspective of NA, but the principle is equally applicable to NB.

NA has to do some preparation work before a negotiation process begins. At first, NA may need to get instructions from his/her supervisor(s) about the negotiation. Instructions may be general ones, such as CA's negotiation policies, or specific ones, such as appropriate negotiation strategies to be used. NA may also need to know CA's current status and relevant negotiation history. The depth of the study on CA depends on many factors, such as the size of the company and the nature of the negotiation. After a careful study, even an experienced negotiator may be surprised to uncover information previously unknown to him/her. Next, NA needs to study NB and CB. Almost any information about the counterpart is useful. The personality and negotiation style of NB would affect the actions taken by NA. Background information about CB helps NA to estimate CB's strengths and weaknesses. Information about the counterpart is usually difficult to obtain, but an experienced negotiator manages to get as much useful information as possible.

NA bargains with NB after the preparation has been done. The bargaining usually starts with the presentation of requirements and facts. Then both sides enter the "give-and-take" stage in which both NA and NB give up something and take something back in return. In general, neither NA nor NB has the full control of the negotiation contents. Sometimes, NA and NB may talk about something totally unrelated to the negotiation, such as sports. Although both NA and NB can speak simultaneously, it is usually the case that one side speaks and the other side listens. Messages are almost always exchanged in an orderly fashion. Bargaining is the main phase of the negotiation process. In fact, it is common to use the terms "negotiation" and "bargaining" interchangeably.

After the bargaining ends with an agreement or a disagreement, NA needs to analyze the result of the negotiation. Lessons learned from a negotiation provide useful guidelines for future negotiations. Some people [SHE99] call this phase "post-negotiation analysis."

3.1.2 Automated Negotiation Process

Automated negotiation makes use of computer networks, especially the Internet, to conduct a negotiation process. Automation itself does not change the nature of the negotiation problem. Preparation, bargaining, and post-negotiation analysis are still the main phases in automated negotiations. Detailed discussion about the negotiation decision process can be found in Chapter 5. Detailed discussion about negotiation phases and their functions can be found in Chapter 6. Figure 3.1 shows an overview of an automated negotiation process. Before a negotiation begins, potential sellers can publish information about the products and services on their websites. Through searching and browsing, or by using an available trading partner discovery mechanism such as the Universal Description, Discovery and Integration (UDDI, www.uddi.org), a buyer can identify potential sellers with whom the buyer wants to conduct negotiations. In order for buyers and sellers to make use of the automated negotiation service, they must each register with a negotiation server of their choice and provide the knowledge that is necessary to conduct an automated negotiation. In Figure 3.1, the registration and knowledge acquisition activity is shown as “a1.”

After a buyer decides to negotiate with a particular seller, the buyer starts the process by submitting an initial product or service request to his/her negotiation server. The negotiation server A (NSA) initiates a bargaining process by creating and sending a Call-For-Proposal (CFP) to NSB, which represents the seller. NSB checks the information and constraints specified in the CFP against the registered information of the seller to determine if there are conflicts in requirements and constraints. The conflicting data conditions and constraints can be modified either by using the registered information and knowledge of NSB or through human intervention to produce a proposal to NSA. The proposal will then be evaluated by NSA against the registered information.

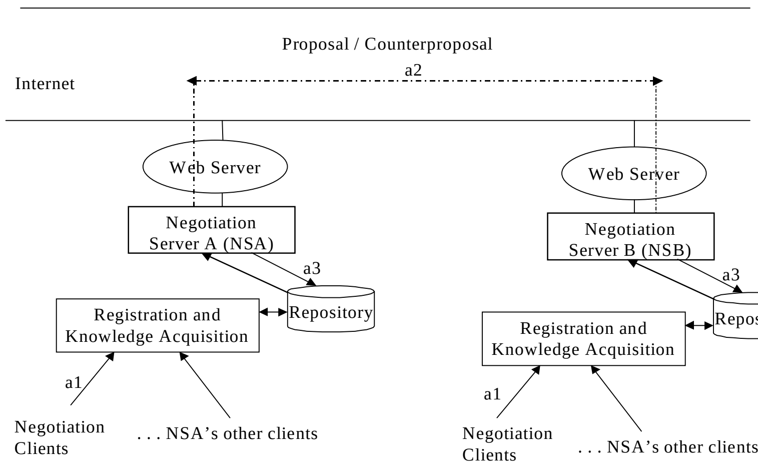
Proposal / Counterproposal
a2
Internet
Web Server
Negotiation Server A (NSA)
a3
Repository
Registration and Knowledge Acquisition
a1
Negotiation Clients
. . . NSA's other clients
Web Server
Negotiation Server B (NSB)
a3
Repository
Registration and Knowledge Acquisition
a1
Negotiation Clients
. . . NSA's other clients

Figure 3.1 Overview of an Automated Negotiation Process

In the event that a conflict or constraint violation is found, relevant rules, which have been provided by negotiation clients, are applied to either (1) reject the proposal and suggest possible changes to NSB, (2) call for human intervention, or (3) generate a counterproposal to be returned to NSB. In the event that the proposal contains a number of alternative data conditions, a cost-benefit analysis component is called to pick the best alternative. This alternative is then sent back to NSB, together with NSA's constraints, as the counterproposal, which starts another round of negotiation. The counterproposal is evaluated in the same way by NSB against its client's registered requirements and constraints. This process will continue until an agreement is reached or either side terminates the negotiation process. The exchange of proposal/counterproposal is shown as “a2” in Figure 3.1. After the bargaining process is finished, information is stored in the repository, shown as “a3” in Figure 3.1.

Activities a1, a2, and a3 in Figure 3.1 roughly correspond, respectively, to the preparation phase, bargaining phase, and post-negotiation analysis phase discussed in the human-based negotiation. The above informal description and Figure 3.1 should help readers to understand the problems to be solved. However, a formal model is needed for the development of an automated negotiation system. The next section addresses this issue.

3.2 A Model of Automated E-business Negotiation

In this section, we present a general model of automated e-business negotiation that captures the key concepts and elements involved in automated negotiations. The model is an abstraction of an automated negotiation system. It is intended to serve as the framework for research and development efforts in this area.

We define the model of automated e-business negotiation as a 6-tuple <C, AN, M, PT, RC, 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 behalf of

clients. The role of the User is to 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 as agents in an agent architectural environment to conduct automated negotiations on behalf of their clients. The negotiation server developed in our earlier work 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 used by the receiver to react to the messages. In our previous work [SU00b], eight primitive operations (Call-For Proposal, Propose Proposal, Accept Proposal, Reject Proposal, Modify Proposal, Withdraw Proposal, Terminate Negotiation, and Acknowledge Message) were introduced. Messages exchanged between replicated negotiation servers are wrapped in a set of XML business object documents (BODs) that 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 [LI01].

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

5. RC stands for a Requirement and Constraint specification model. A data model is needed to specify the requirements and constraints associated with a product/service that 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. The current version of the negotiation server, which has been implemented at DSRDC of UF, uses an Active Object Model (AOM) for requirement and constraint specifications [LEE00].

6. DM stands for a Decision Model. As we have pointed out in Chapter 1, 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 implemented negotiation systems, our negotiation server implements several rules of concessions that 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 decision model (DM) has become our main focus. We will present the results of our research in Chapter 5.

Various negotiation models have been proposed for different purposes. A brief overview of these negotiation models is presented here. Ben-shaul et al. [BEN98] described a negotiation model for the dynamic composition of distributed applications that are built by selecting, deploying, and invoking components from remote sites. Suppose site1 needs to build a distributed application, but site1 does not have all the components needed for the application. Therefore, site1 may need to send requests to other sites, such as site2 and site3, asking for components. Since most components are from other sites, it is necessary for site1 to "negotiate" with site2 and site3 about the terms and conditions of using their components. The negotiation model in the work has two independent layers: the generic negotiation layer and the specific resource-allocation layer. The generic negotiation layer is based on mediated negotiations, where a mediator is in charge of conducting the negotiation process. Several negotiation strategies controlling the negotiator's behavior are defined: the initial offer, the improvement strategy, when to accept an offer, and when to quit the negotiation. In our opinion, DM in our negotiation model covers negotiation strategies discussed in the paper. The resource-allocation layer is based on a rental model that has three additional attributes to be negotiated: permissions, expiration time, and renewal period.

Kang and Lee [KAN98] discussed a negotiation model with multiple negotiation attributes (called "transaction factors" in the paper) and learning capabilities. Negotiation attributes are presented as a hierarchy of an ontology, having category-independent attributes and category-dependent attributes. The learning capabilities are shown by the mathematical formula for price, and a rule-based knowledge system for attributes other than price. Our negotiation model does not include an ontology element because it is assumed that both negotiators use a common business vocabulary. However, we realize that ontology is an important research issue for e-business negotiation, especially in the early contact stage among business enterprises.

Jennings et al. [JEN00] discussed automated negotiation. Although the paper does not explicitly introduce a negotiation model, it proposes three broad research topics for automated negotiation: negotiation protocols, negotiation objects, and agents' decision making models. "Negotiation protocols" in the paper also include participants who are C (clients) in our negotiation model. "Negotiation objects" in the paper correspond to M (negotiation messages) in our negotiation model.

Some existing work surveyed in Chapter 2 includes negotiation models in specific application domains. Sierra et al. [SIE97] proposed a service-oriented negotiation model between autonomous agents. The model deals with issues such as how to generate the initial proposal, how to evaluate incoming proposals/counterproposals, and how to generate counterproposals. In our opinion, the model is essentially a subset of DM in our model of automated e-business negotiations. ICE standard proposed an ICE negotiation model for managing the syndicator/subscriber relationship. Unlike our negotiation model, which is applicable to most types of e-business negotiations, the ICE negotiation model is domain-specific. Some elements in our negotiation model are either missing or assumed to be well-known in the ICE negotiation model.

CHAPTER 4
NEGOTIATION PRIMITIVE AND PROTOCOL

Two important issues are discussed in this chapter: negotiation primitives and the negotiation protocol. Negotiation primitives are meaningful message types to be exchanged between two negotiation servers. They are the basis for composing negotiation messages, which are physical implementations of negotiation primitives in some data structures. The negotiation protocol, defined in the form of a state transition diagram, governs the execution of a negotiation server. Most of the negotiation decision-making tasks to be discussed in Chapter 5 are actually the conditions for transiting from one state to another.

4.1 Negotiation Primitives

Eight primitive operations for negotiation have been introduced in our previous work [HUA00]. They are: CallFor Proposal (CFP), Propose Proposal, Reject Proposal, Withdraw Proposal, Accept Proposal, Modify Proposal, Acknowledge Message, and Terminate Negotiation. Table 4.1 gives a brief description for each primitive.

CallFor Proposal and Process Proposal are alternative ways of initiating a negotiation transaction. Accept Proposal and Terminate Negotiation are alternative terminators that can bring a negotiation transaction to an end. Acknowledge Message is used to acknowledge the receipt of a message. The remaining primitives, together with Process Proposal, are iterators that can be used repeatedly in a negotiation process to do bargaining. Since a negotiation party is allowed to proactively propose an offer to the other side, Propose Proposal is used as both an initiator and an iterator. The situation is similar to the one where an unsolicited proposal is presented to a buyer from a supplier or vice versa. The conceptual state transition using these negotiation primitives is not complicated. In essence, a negotiation transaction is an "initiation-iteration-termination" procedure. The actual implementation, however, involves many details to be considered. Section 4.2 will describe in detail the state transition diagrams that define the negotiation protocol.

Table 4.1 Negotiation Primitives

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

4.2 Negotiation Protocol

To ensure effective and meaningful communications between negotiation servers, they must follow a well-defined protocol to exchange negotiation messages in a negotiation process. For instance, after the buyer's negotiation server sends out a CFP, it expects to receive a Propose Proposal from the negotiation partner. Other types of messages would not be meaningful. Several negotiation protocols for bidding and other types of auctions are available and have been used in some automated systems. In this section, we use two state transition diagrams to define a bilateral bargaining protocol. Figure 4.1 defines the states and state transitions of the buyer, and Figure 4.2 defines those of the supplier.

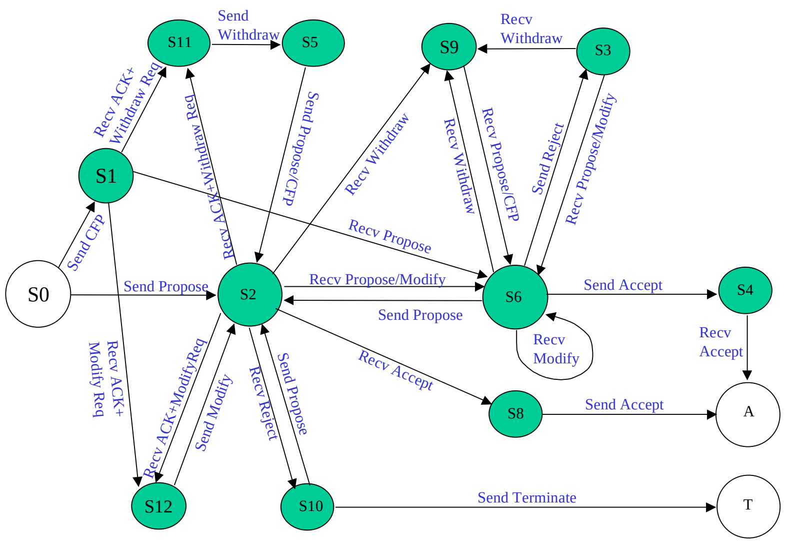

Figure 4.1 Buyer's Negotiation Protocol and State Transition Diagram

There are a total of 15 different negotiation states that a negotiation server can be in during a bilateral bargaining process. Since a negotiation server can be the initiator of a negotiation transaction and, at the same time, the negotiation partner of a transaction initiated by another server, the transition diagrams define the various states that a server can be in when playing both roles. We shall use “buyer” and “supplier” to represent these two roles. The meanings of these states are shown in Table 4.2.

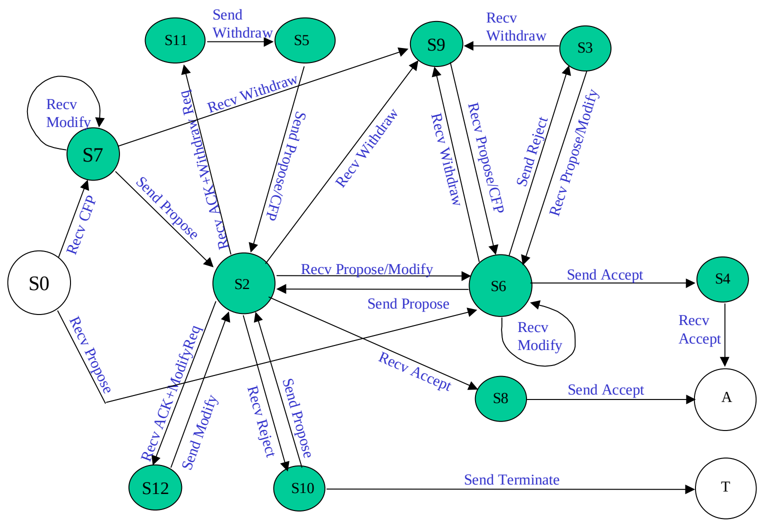

Figure 4.2 Supplier’s Negotiation Protocol and State Transition Diagram

Table 4.2 Negotiation State Semantics

S0	Initial state
S1	CFP is sent
S2	Propose is sent
S3	Reject is sent
S4	Accept is sent
S5	Withdraw is sent

Table 4.2 – continued.

S6	Propose/CFP/Modify is received
S7	CFP/Modify is received
S8	Accept is received
S9	Withdraw is received
S10	Reject is received
S11	Received the Acknowledge of the Propose/CFP that was sent and the user requests the Withdrawal of that Propose/CFP
S12	Received the Acknowledge of the Propose/CFP that was sent and the user requests the Modification of that Propose/CFP
A	Agreement is reached
T	Negotiation is unilaterally terminated

To simplify the explanation of Table 4.2, we shall use the verbs of negotiation primitives instead of their corresponding negotiation messages, which are actually transmitted between negotiation servers. S0 is the initial state. S1 is entered after the buyer sends out a CallFor. S7 is entered when the supplier receives a CallFor. The remaining states are shared by both the buyer and the supplier. In general, a negotiation transaction is initiated by the buyer after sending out a CallFor. In the CallFor, the buyer indicates his/her interest in starting a negotiation on some negotiation item(s) and provides some constraints. To respond to a CallFor, the supplier would send a Propose (i.e., proposal) and provide the negotiation conditions and constraints from his/her own perspective. Then the buyer can decide whether the constraints in the proposal are acceptable or not. If they are not acceptable, further negotiation needs to be conducted to resolve the differences by sending a Propose (i.e., counterproposal) to the supplier. In another case, if the buyer knows precisely the constraints of a negotiation item he/she wants to send to a supplier, it can use Propose to start the negotiation instead of CallFor. Hence, the supplier may also receive a Propose to start a negotiation. Either case (start by sending either CallFor or Propose) is defined by the messages and state transitions surrounding S0.

The most frequently used primitive is Propose, which transmits a proposal or counterproposal between two negotiation servers. S2 and S6 are two states of a negotiation server when it sends and receives Propose, respectively. To respond to a Propose, several messages are allowed: another Propose to issue a counterproposal, a Reject to indicate the rejection of some conditions specified in the original proposal, and an Accept to indicate the acceptance of the terms and conditions specified in the previous proposal without further modifications. Note that in our negotiation protocol, a Reject message is not for specifying the termination of the negotiation. Rather, it is used to convey the dissatisfaction of some conditions to the negotiation partner, and for the partner to modify a proposal and send it back. Thus, the Reject contains the rejected conditions. For a negotiation server to unilaterally terminate a negotiation process for any reason, the Terminate can be used. The state transitions and messages among states S2, S10, and T describe the cases of rejection and termination. In fact, at any state except the initial state S0, if a Terminate is received or sent, the negotiation state would transit to T and the negotiation process terminates. However, in order not to clutter the diagram, the state transitions to T are not shown in the figures.

To reach an agreement in a bilateral negotiation, both sides exchange a pair of Accept messages containing identical conditions that made the deal. This is the same as putting both signatures on an agreed contract. It allows both sides to ensure that their counterparts have accepted the conditions as specified. The messages from S2 to S8 and from S8 to state A illustrate this case. A simplified alternative is for the receiver of an Accept message to simply return an acknowledgment instead of the second Accept.

4.3 Negotiation Scenarios

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

(1) A counterproposal scenario:

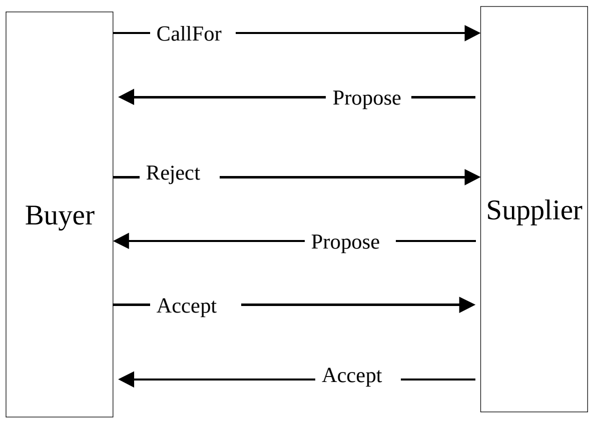

Figure 4.3 Counterproposal Scenario

Figure 4.3 shows that the buyer sends a CallFor to the supplier and the supplier sends a Propose back. The buyer is not satisfied with the conditions, so the buyer sends a Reject to the supplier. After the supplier receives a Reject, the supplier returns a modified proposal (Propose) back to the buyer (Buyer State: S0->S1->S6->S3->S6, Supplier State: S0->S7->S2->S10->S2). At this time, the buyer is willing to accept the modified proposal (Buyer State: S6->S4->A, Supplier State: S2->8->A). The supplier confirms the acceptance from the buyer by sending an Accept to the buyer.

(2) A scenario with Modify/Withdraw message exchange:

Sometimes a negotiator wants to change some conditions in the previous proposal or withdraw the proposal that already has been sent. Two primitives are provided: Modify and Withdraw. When a negotiation server receives a Modify or Withdraw message, it should stop the processing of the previous Propose message. In the case of Modify, the negotiation server needs to combine the contents of the previous proposal with the modifications specified in the message for Modify to produce a new proposal. In the case of Withdraw, the negotiation server will receive a new proposal from the sender of Withdraw. Both cases require the negotiation server to respond based on the new proposal. As described in the state transition diagram, Modify or Withdraw for the CallFor message is similar to the case of Propose. Moreover, since the modification and withdrawal of a Propose or a CallFor may require human intervention, two human input actions, ModifyReq (for modification request) and WithdrawReq (for withdraw request), are also introduced in the protocol state diagrams. A scenario that involves Modify/Withdraw is shown in Figure 4.4.

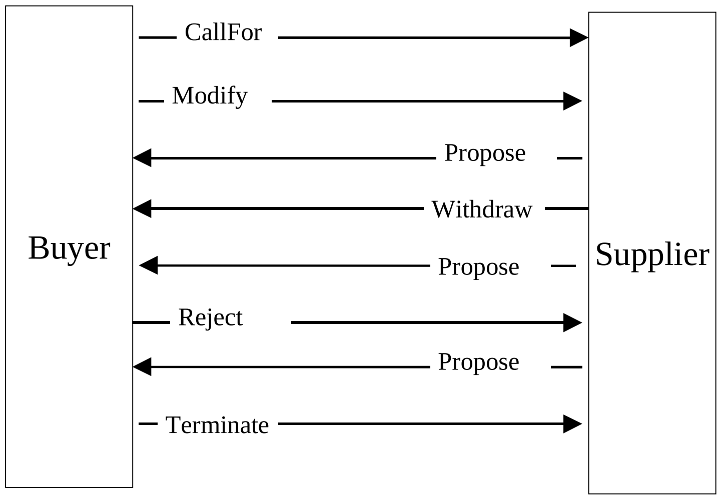

Figure 4.4 Withdraw/Modify Scenario

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

The above scenario is an uncomplicated case of using Modify/Withdraw in a negotiation. Unlike other negotiation primitives, which are designed to respond to messages received from the negotiation partner, Modify and Withdraw primitives are designed to change the proposal that was sent out by a negotiation server. It is possible that this may cause synchronization problems between the two negotiation servers. If we refer to the negotiation server that sends a Modify/Withdraw as S and the negotiation server that receives a Modify/Withdraw as R, it is possible that R may have responded to the previous proposal before it receives the Modify/Withdraw message (Synchronization Problem 1). To solve this problem, as shown in Figures 4.1 and 4.2, several state transitions have been added to the state transition diagrams so that the R of Modify/Withdraw can reverse the state to process Modify/Withdraw, even if it has responded to the old proposal. Correspondingly, S needs to discard the obsolete message from R and wait for a new message. This can be achieved by letting the negotiation server transit to some states (e.g., S5, S11, S12 of Figures 4.2 and 4.3) in which all the incoming messages will be ignored (including the obsolete response message for the old proposal) after it receives the Modify/Withdraw request from a human.

In another case, the negotiation servers S and R may send a Modify/Withdraw message at the same time after S sent a proposal and R returned a counterproposal. However, it is obvious that only S’s Modify/Withdraw can take effect because R’s counterproposal has been invalidated by S’s Modify/Withdraw. Therefore, the Modify/Withdraw of R’s counterproposal also should be treated as invalid (Synchronization Problem 2). To solve this problem, we include a mechanism to prevent both sides from sending Modify/Withdraw at the same time. In other words, Modify/Withdraw is similar to an exclusive token that only one negotiation server can use at any time. The mechanism works as follows. First, after a negotiation server receives a Propose, an Acknowledge message is returned to the sender if the receiver decides to process instead of ignore the Propose message. In order to simplify the diagrams given in Figures 4.1 and 4.2, we did not show the sending and receiving of Acknowledge messages. Second, a negotiation server can send a Modify or Withdraw message only after its proposal has been acknowledged. In this case, after the user has begun the Modify/Withdraw process by sending a ModifyReq/WithdrawReq to the negotiation server, all the incoming obsolete messages will not be acknowledged. Therefore, the negotiation partner cannot send its Modify/Withdraw message before it processes the received Modify/Withdraw message.

CHAPTER 5
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 7-tuple <W, CX, G, L, P, S, CB> 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 an enterprise.

4. L is a set of plans of decision and actions used by an AN.

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.

7. CB 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 also needs to evaluate and rank available options and select the proper one to form the counterproposal. In the existing negotiation server, we have adopted the CB proposed by Su et al. [SU87] and have implemented several proposed preference scoring and aggregation methods.

Figure 5.1 shows the inter-relationship of these key concepts except CB, which is used inside L. The figure will be explained progressively in the following sections.

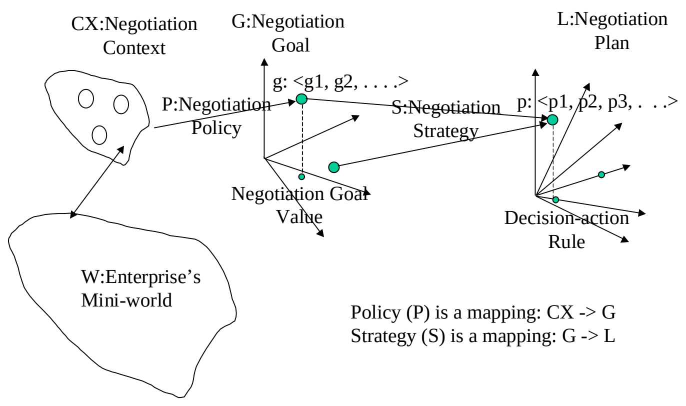

Figure 5.1 Key Concepts in the Decision Model

5.1 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 information and material, financial, and personnel resources that exist in the real world. These resources may be available either in-house or in other enterprises, but are accessible by the enterprise. They represent the internal and external conditions and states of the enterprise's business. The mini-world of one enterprise can be expected to be quite different from those of others. It 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 level, 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 5.1). These contextual expressions capture the typical negotiation conditions and situations that the enterprise encounters and for which negotiation goals can be specified.

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 1) a small company without a good credit and 2) 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 or even no 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. 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 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 5.2. 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 to 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 a 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 can be 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. Some people may prefer to use Cost (C) for the buyer side. Since Profitability and Cost are two different ways to specify the same concept, we will use Profitability (P) throughout this dissertation.

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 a deadline is missed, 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 and deliver 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 maker 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 startup 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 triple (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.

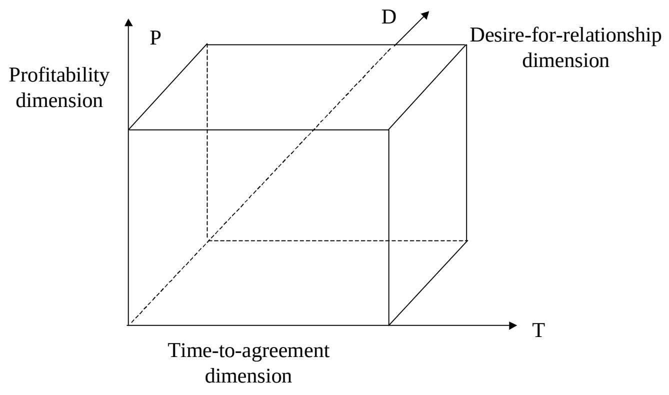
P
Profitability dimension
D
Desire-for-relationship dimension
T
Time-to-agreement dimension

Figure 5.2 Example of a 3-dimensional Goal Space

5.2 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 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 is a course of action, guiding principle, or procedure considered expedient, prudent, or advantageous. One example is Honesty is the best policy. The third 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 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 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. Therefore, 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 those goals. The specifications of decisions and actions for implementing negotiation policies are called negotiation strategies. We will address negotiation strategies in Section 5.5.

Before we continue our discussion on negotiation policy, we need 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 that 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 both security polices with a role-based access control and general purpose management policies. An application of Ponder for realizing enterprise viewpoint concepts is reported in Lupu et al. [LUP00].

A Routing Policy Specification Language (RPSL) [ROU99] is described in a Request For Comments (RFC) of the Internet Engineering Task Force (IETF). 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 those policies and domains. While SPSL currently supports policies for packet filtering, IP Security (IPsec), and Internet Key Exchange (IKE), 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-business.

Negotiation policies are specifications that 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 5.1. A specific policy can thus be specified by the general expression: If CXi then Gj, where CXi is a member of CX and Gj 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 company's crediting rating is excellent) 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 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 5.1 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 following three subsections.

5.3 Decision-Action Rule

As described in Section 4.2, a negotiation protocol can be conveniently defined by a finite state diagram consisting of a number of states and transitions. At each state, an AN needs to make a decision and to take some actions 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 Figures 4.1 and 4.2), 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 bilateral negotiation protocol given in Chapter 4, the transitions in various states of the protocol are shown below:

1. The initiation of a negotiation transaction (issuing the initial negotiation proposal or call-for-proposal)
2. The acceptance of a proposal
3. The rejection of a proposal
4. The termination of a negotiation transaction
5. The modification of a proposal
6. The withdrawal of a proposal
7. The generation of a counterproposal

In the following subsections (Subsections 5.3.1 to 5.3.7), we will 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. Decision-action rules capture the techniques used by skilled human negotiators and are used by an AN to conduct the negotiation on behalf of the enterprise.

5.3.1 Initiation of a Negotiation Transaction

In the current negotiation protocol, a negotiation transaction 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., close to 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 quite familiar with the counterpart or 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 from the counterpart.

5.3.2 Acceptance of a Proposal

Let us 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 analyzing its contents, some decision-action rule needs to be applied to guide the acceptance decision. Some examples are given below:

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 closer to the negotiator's position. On the other hand, if X is a large value, the negotiation plan is a cooperative one. This is because the negotiator is willing to accept a counterpart's offer that is not very close to the negotiator's desired value. Different X values used in the parameterized rules (a and c) represent different rules.

5.3.3 Termination of a Negotiation Transaction

The negotiation protocol defined and used in the current 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 because the negotiation parties may engage in an endless exchange of proposals. To ensure a negotiation transaction’s convergence on 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 that have been 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 or is not acting reasonably. 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 X number of times.

b. The global preference scores derived for the received proposals 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.

5.3.4 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. A rejection simply informs the supplier's negotiation server that the received proposal is not acceptable and requests the sender to modify and re-submit 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 concede further.

5.3.5 Modification of a Proposal

During the negotiation process, the User, who is monitoring the progress 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 modifications 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, or other details specified in the previous proposal. If the inventory has dropped to a certain level, there may be a need to change the quantity to be negotiated.

5.3.6 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 to make the withdrawal decision automatically. For example, if the financial condition of the issuer has changed, the customer may change his/her order. In a company in which prices of products to be sold are updated monthly or quarterly, the negotiator may need to withdraw the proposal that has been sent if no response has been received and the negotiation time has reached the monthly or quarterly boundary.

5.3.7 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 (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. 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 the highest global cost-benefit score)

b. Purely unselfish (i.e., select the one with the lowest global cost-benefit score)

c. Some point in between.

In generating a counterproposal, a negotiation server may have to concede 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.

5.3.7.1 Static Concession

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

One useful function is the sigmoid function [VON93]: f(x) = 1 / (1 + exp (- g * (x - 4))). The sigmoid function results for g = 1.5 are shown in Figure 5.3.

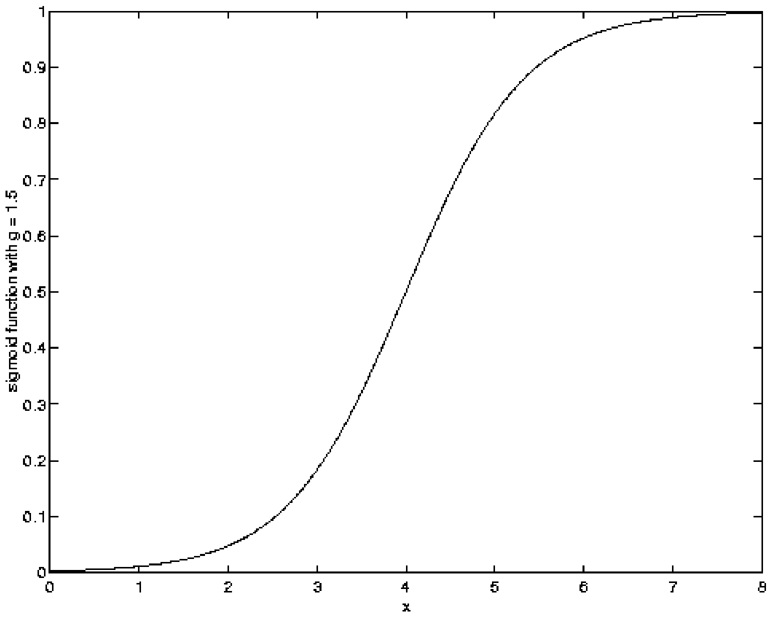

Figure 5.3 Graph for Sigmoid Function with g = 1.5

As a decision-action rule, this function can be used to control the rate of concession. As shown in the graph, the concession speed is initially very slow. It increases dramatically in the middle of the negotiation. Then the concession speed 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 5.5), the concession speed can be controlled. By controlling the concession speed, the speed of convergence of a negotiation transaction can also be controlled (e.g., the higher the concession speed, the faster the negotiation can reach an agreement). The curve has one turning point (when x = 4). It is possible to define functions that have multiple turning points.

Again, alternative parameter values of a parameterized decision-action rule (in this case "g" values) 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 to give no concession at all under certain situations. This phenomenon is called Boulwarism [THO98]. Users can also define arbitrary functions based on user-selected pivotal points and interpolation and extrapolation techniques.

5.3.7.2 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. That is to say, the function can be adaptive. Two examples of dynamic concession strategies are given 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) = [A * own_concession(x)] + [(1.0 - A) * counterpart_concession(x)]

where A is the adjustment factor whose value is in the range of 0.0 to 1.0, own_concession (x) is the function to compute the concession used by my own side, and counterpart_concession (x) is the function to compute the concession used by the counterpart.

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

There are several ways to compute the counterpart_concession (x). 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 (1, 20.0), (2, 17.0), (3, 15.0), and (4, 12.0) in the X-Y coordinates, then compute coefficients for a 4-degree polynomial function. 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 my-own-side but important to the counterpart, and decrease the concession speed of those attributes that are important to my own side but not important to the counterpart.

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.

5.4 Plan of Decision and Action

If D1, D2, . . . Dn (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 enterprise can be represented by a set of tuples having the general structure <d1, d2, . . . di, . . . dn> where di in Di 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 5.1. 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 prove to be.

5.5 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 American Heritage Dictionary of the English Language, Third Edition [AME96]. They are shown below:

1. The science and art of using all the forces of a nation to execute approved plans as effectively as possible during peace or war.
2. The science and art of military command as applied to the overall planning and conduct of large-scale combat operations.
3. A plan of action resulting from strategy or intended to accomplish a specific goal.
4. 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 5.1 and negotiation plans in Section 5.4, we can formally define a negotiation strategy as a function S: G -> L, where G is a set of goals in the goal space, L is a set of negotiation plans in the plan space, and S maps from G to L. Conceptually, strategies are mappings from certain points in the negotiation goal space to certain points in the negotiation plan space, as illustrated in Figure 5.1. Each mapping can be either one-to-one, many-to-one, many-to-many, or one-to-many. In the case of a one-to-one mapping, a person who plays the role of Negotiation Expert in a business enterprise has identified a specific goal and has a specific plan of decisions or actions that can be used by a negotiation server to achieve that goal. In the case of a many-to-one mapping, the expert specifies that a number of goals can be achieved by a single negotiation plan. For example, a strategy specification may state that if P dimension in the goal space is in the range of (0.2 to 0.5), T in the range of (0.6 to 0.7), and D in the range of (0.6 to 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, and so on. In effect, the range specifications for P, T, and D allow a number of goals to be mapped to a specific negotiation plan. In the case of a many-to-many mapping, multiple negotiation plans can be applied to achieve multiple negotiation goals. In the case of a one-to-many mapping, a goal can be realized by multiple plans. This represents the case where 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 that mini-world. A particular goal will be selected based on a specific policy that 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 plan 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, and other actions. 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.

5.6 Chapter Summary

We will now summarize what we have presented in this chapter. Referring back to Figure 5.1, 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 specifications of particular plans of decisions or actions (to be taken at various steps of a negotiation protocol) that are needed to achieve the goals (i.e., negotiation strategies map goals to plans of decisions and actions). Negotiation strategies can be defined by people who play the role of Negotiation Experts in the form of strategic 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 negotiation 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 a 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 vice versa. The separation of policy mapping between negotiation contexts and goals and strategy mapping between goals and plans gives 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 designers of policy and strategy a finer granularity of policy and strategy specifications.

To relate what we have presented in this chapter with some existing works, we note that the term "negotiation strategies or tactics" used in the existing work we surveyed corresponds to our decision-action rules. It forms a subset of the decision-action rules suggested in this dissertation. 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 “counterproposal generation” 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, nor high-level negotiation policies. Sierra et al. [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 formulae for tactics, which correspond to decision-action rules in our work. It is pointed out by Sierra et al. 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.

CHAPTER 6
NEGOTIATION LIFE CYCLE

The concept of life cycle is not new. Life cycles have been applied to product development and software development for years. Software development life cycles go through phases of requirements analysis, design, implementation, testing, and deployment. Computer-Aided Software Engineering (CASE) tools are programs that aid or automate these phases. Software development in the 1950s and 1960s 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 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 by following a hand-coded negotiation protocol and hand-coded negotiation strategies. The analysis of negotiation contexts, goals, policies, product/service requirements and constraints, the design of negotiation strategies, plans of decisions or actions, product/service evaluation methods, and the like, have not been incorporated into 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 study is represented by two models, which will 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. Our work also proposes three negotiation roles, each played by different people.

6.1 Review of Existing Work on Negotiation Life Cycle

Three phases are discussed in the life cycle presented by Robinson and Volkov [ROB98], as shown in Figure 6.1: 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 phase is to plan for achieving the negotiation goals through interactions with the counterparts using appropriate techniques. The task of the negotiation implementation phase 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 relatively 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 negotiation 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 can extend this model in two ways. First, we understand that reaching either the agreement state or 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 weaknesses in the negotiation policy and strategy. If a negotiation is terminated unsuccessfully, we need to determine what factors contributed to the failure. Is it because the price was too high (for sellers) or too low (for buyers)? Is it because the delivery date was 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 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, optionally, post-settlement, as discussed in Section 2.4.

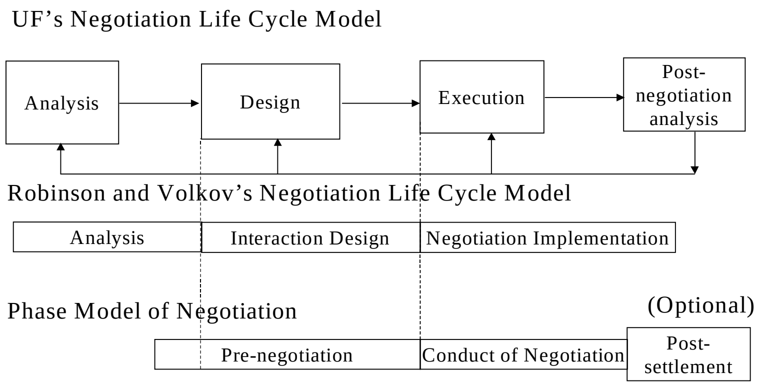
UF’s Negotiation Life Cycle Model
Analysis
Design
Execution
Post-negotiation analysis
Robinson and Volkov’s Negotiation Life Cycle Model
Analysis
Interaction Design
Negotiation Implementation
Phase Model of Negotiation
(Optional)
Pre-negotiation
Conduct of Negotiation
Post-settlement

Figure 6.1 UF’s Negotiation Life Cycle Model and Its Relationship with Two Existing Models

6.2 Negotiation Life Cycle Model for Automated E-business Negotiation

We combine the phases of the above two models and introduce a four-phased negotiation life cycle model, as shown in Figure 6.1. The figure also shows how our four phases relate to those of the two models. Our model makes use of the concepts presented in the previous chapters of this dissertation. The 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 and 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 capturing 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. This phase makes use of 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. Among all the phases in our model, this phase is the most observable one because it deals directly with the negotiation counterpart. As a result, existing research on automated e-business negotiation mainly focuses on this phase. Although it is vital to have a good execution plan, it is equally important to do the preparation work before going to the "electronic bargaining table." That is the main reason why two phases are introduced before the execution phase.

The post-negotiation analysis phase takes the outcome of the execution phase, analyzes it, and provides 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 behaviors of the negotiation server in subsequent negotiation processes.

6.3 Negotiation Roles

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 interacting with the negotiation system during a negotiation process. These roles can be played by individuals or by groups of people. In general, corporate executives would play the role of the Policy Maker. Negotiation practitioners, purchasing managers, and sales managers would play the role of the Negotiation Expert. Sales persons and purchasing agents would play the role of the User. A person may play multiple roles. For example, a corporate executive may be the Policy Maker, and the Negotiation Expert, as well as the User.

Figure 6.1 also shows the approximate correspondence among these models. Our negotiation life cycle model is the only one that includes a feedback mechanism.

Robinson and Volkov’s negotiation life cycle model does not have a phase after negotiation implementation because it focuses mainly on each independent negotiation. Its pre-negotiation phase in NSS deals with preference solicitation and utility construction. It covers only part of our analysis and design phases.

CHAPTER 7
SYSTEM ARCHITECTURE

The system architecture of the current 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. New components for managing negotiation history and statistics and for providing a feedback mechanism to other phases of the proposed negotiation life cycle are also needed. Before presenting the new system architecture, an overview of the current negotiation server architecture is described so that the readers may understand how the additional components introduced in the new system architecture relate to some of the components in the existing architecture.

7.1 System Architecture and Components of the Existing Negotiation Server

The architecture of the rule-based, automated negotiation server developed at the University of Florida under a NIST supported project is described in papers by Huang et al. [HUA00, SU00a, HAM00]. This server can be replicated and installed at multiple sites on the Internet, as illustrated in Figure 7.1. Two negotiation parties conduct a negotiation through a communication infrastructure on the Internet using negotiation primitives and negotiation messages encapsulated in XML BODs. The components of the current negotiation server are shown in Figure 7.2. 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.

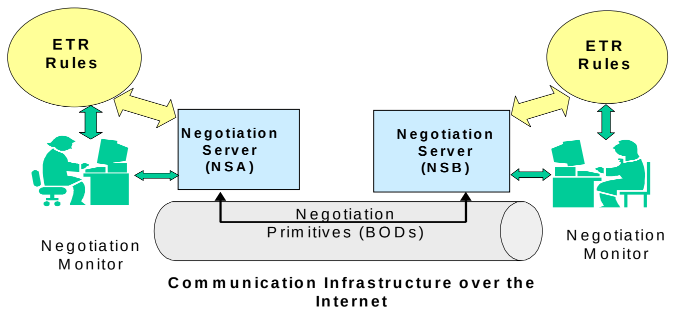

Figure 7.1 Current Negotiation Architecture

(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. CSP 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, it triggers the proper decision-action rules to relax constraints, to inform the user, and so on.

(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 messages used by the Negotiation Server.

(6) Negotiation Messaging: Provides the sender and receiver using 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 forwarded to Web-based Negotiation Console. It also provides a log management for all negotiation transactions for future use.

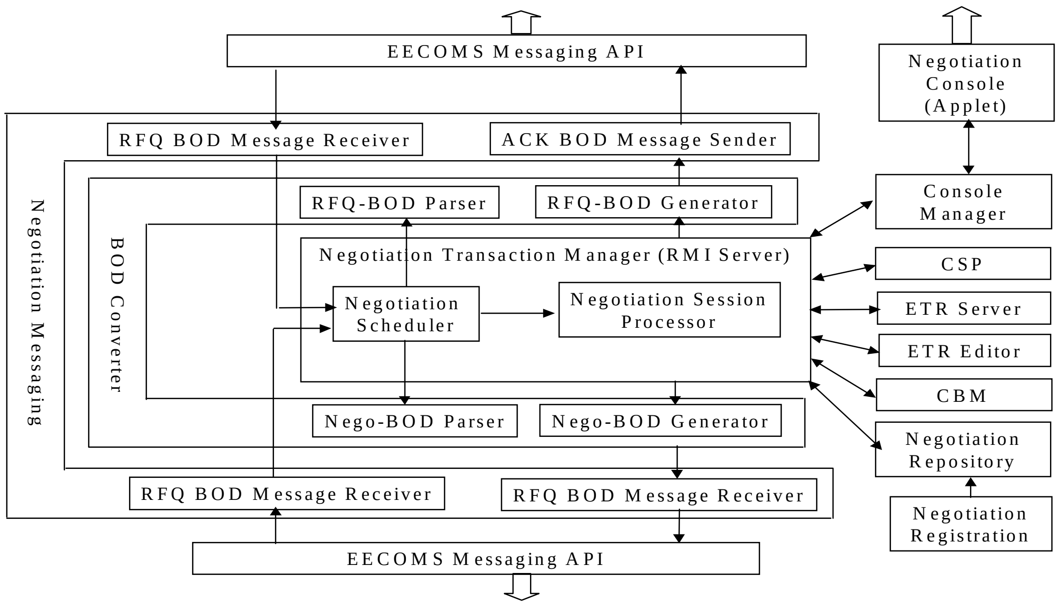

Figure 7.2 Components of the Current Negotiation Server

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, rules, and XML BODs 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 ETR definition at build-time and for dynamic rule editing at run-time to change negotiation strategies dynamically.

7.2 New System Architecture

Figure 7.3 shows the new system architecture. The current negotiation server is shown as a dashed box in the new system architecture. Changes to the current negotiation server are necessary 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 highlighted in gray. 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).

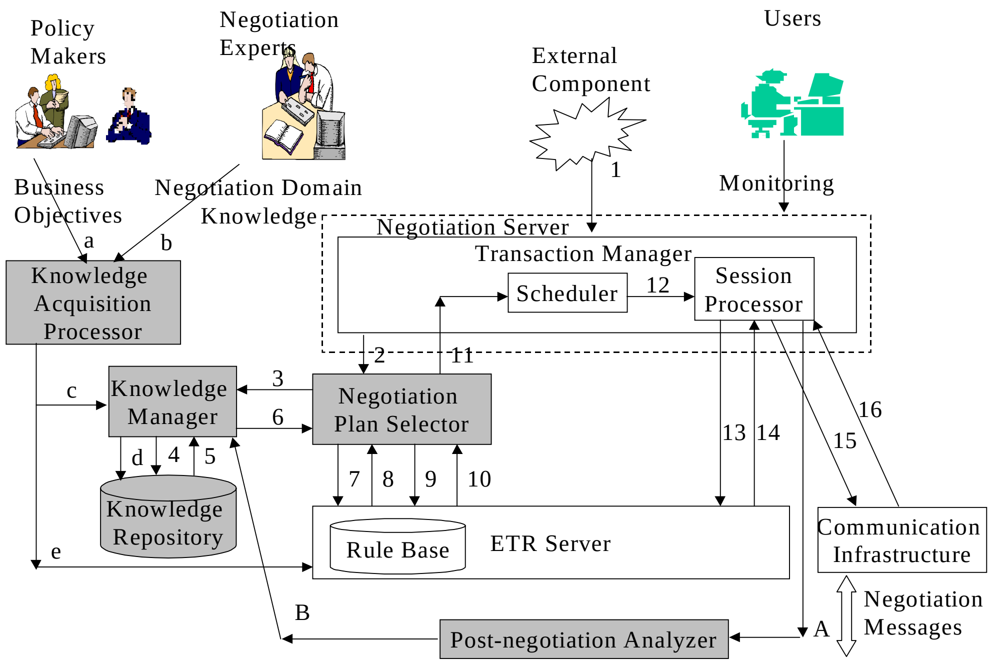

Figure 7.3 New System Architecture

The three negotiation roles discussed in the previous chapter are shown in Figure 7.3. 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 to define negotiation strategies and plans of decisions and 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 statistical information from 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 12 are the per-transaction activities, and activities 13 to 16 are the per-session activities. Since a negotiation transaction may involve multiple negotiation sessions (a session is a transmission of a negotiation message and the related processing), activities 13 to 16 may be executed multiple times; however, activities 1 to 12 are executed only once for each negotiation transaction. The Negotiation Transaction Manager 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 specifications of the product/service that the requester wants to buy/sell). The Negotiation Transaction Manager would then activate the Negotiation Plan Selector (NPS) and ask it to select the proper plan of decisions and 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 receive 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., taking 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 and actions specified by a set of decision-action rules. After the decision-action rules have been selected, the ETR server would notify the NPS (Activity 10), which in turn notifies the Scheduler (Activity 11) 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 12). It would consult with the ETR server for each transition decision (Activity 13) by posting an event to trigger the proper decision-action rule. The ETR server would return the decision result to the Session Processor (Activity 14), which then composes a proper negotiation message based on the decision and sends it through the communication infrastructure (Activity 15) to the counterpart's negotiation server. The Session Processor would then wait for the response of the counterpart's negotiation server (Activity 16). Activities 13 to 16 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 state or termination state. The result of a negotiation transaction and its statistical information (e.g., the number of message exchanges, the time it takes for each session, etc.) are stored in the Knowledge Repository through the Knowledge Manager. The information is used for the future selection of policies, strategies and negotiation plans (Activities A and B of Figure 7.3).

CHAPTER 8
IMPLEMENTATION AND MODEL EVALUATION

A large part of the new system architecture has been implemented by integrating the existing negotiation server with some software components developed at DSRDC of the University of Florida, plus some additional components developed by this author. The author has also conducted several interviews to validate the negotiation models presented in this dissertation.

8.1 Negotiation Messages

In human-based negotiation, messages in natural languages are exchanged among human negotiators. In automated negotiation, it is simpler from the implementation point of view for negotiation servers to exchange messages in a structured format than in natural languages. Also, the message contents should be restricted to issues that are directly relevant to the subject of negotiation. This is quite different from human-based negotiations in which irrelevant matters such as weathers, sports and jokes are discussed to relieve the tension around the bargaining table.

Since business negotiation is viewed as an integral part of business activities, it would be ideal if message exchanges in business negotiation could be represented by a set of e-business negotiation messages (EBNMs). These messages are the implementation of the negotiation primitives discussed in Chapter 4. They are similar to the standard business objects defined by the Open Applications Group (OAG) for other business activities such as Request for Quotation, Process Purchase Order, etc. In this dissertation, we have designed a set of EBNMs following the same data structures and philosophy of OAG's business object documents (BODs). These EBNMs are expressed in XML format for transmission between automated negotiation systems [LI01].

8.1.1 Data Structures of Negotiation Messages

As shown in Section 4.1, each negotiation primitive is composed of a verb and a noun. The verb names the action of the sender and the noun names the information contents to be processed by the receiver. EBNMs are the formal specifications of these primitives. In this section, we focus on the information contents that go with the verbs: CALLFOR, PROPOSE, ACCEPT, MODIFY, TERMINATE, WITHDRAW, REJECT, and ACKNOWLEGE. We distinguish two general types of information contents in the formal specification: attributes and constraints.

8.1.1.1 Attributes

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

8.1.1.2 Constraints

Constraints, such as those described by Mariott and Stuckey [MAR98] and Tsang [TSA93] are widely used in computer science. Constraints associated with negotiated items can be specified in a negotiation proposal. Three general types of constraints are distinguished: attribute constraints, inter-attribute constraints, and inter-item constraints. An attribute constraint specifies the constraint on the value of an attribute. For example, the memory size must be greater than 64M bytes, or is equal to one of the enumerated values, or falls in a specified value range. An inter-attribute constraint specifies the relationship among a number of attributes and values. For example, a constraint may state that "If the model of the computer is Pentium II 500 and the disk capacity is greater than 10 GB, then the memory size has to be greater than 64MB." Inter-item constraints are used to specify the relationship among negotiation items specified in a negotiation proposal. An example would be "If I cannot get the same number of Ethernet cards to go with the number of computers I am ordering, I do not want either of these items," meaning the quantities of these two items have to be the same. In our design of EBNMs, attribute constraints are specified together with attributes, and inter-attribute constraints are specified at the level of a data structure above the specification of both item attributes and transactional attributes, because the constraints may involve both types of attributes and values. Inter-item constraints are specified at the level above the specification of negotiation items for obvious reasons.

The above three general types of constraints can be specified in different constraint languages with different syntaxes. Different application systems and users may have their own preferences for the languages they use. For that reason, it is important to leave the languages open. In the design of negotiation messages, constraint specifications are values of some general attributes. Thus, the interpretation of the constraint specifications is left to the application system that receives an EBNM. A new field called ConstraintLanguage, whose value names a constraint specification language, is added to a common header of an EBNM. UFCL, a constraint specification language developed by Huang [HUA00], is used in our implementation.

8.1.2 Design of Business Negotiation Messages

The following notation similar to Extended Backus-Naur Form (EBNF) and XML DTD is used to describe the structures of EBNMs: Negotiation_Message_Name (Substructure1, Substructure2+, Substructure3*, Substructure4? . . .). Substructures are defined in terms of their own underlying structures. The process repeats until the substructure is of a string type or numeric type. Symbols like +, *, and ? at the end of each structure represent one or many, zero or many, zero or one (i.e., optional) occurrences, respectively.

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

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

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

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

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

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

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

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

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

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

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

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

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

```
TERMINATE_NEGOTIATION (HEADER, NEGTRANSACT)
NEGTRANSACT (NEGTRANSACTID)
```

The EBNM for Reject Proposal tells the negotiation partner that the previous proposal is not acceptable. However, the sender does not want to terminate the negotiation and is willing to accept a modified proposal from the receiver. This negotiation message may optionally give the reason for rejection by pointing out the attributes and constraints in the received proposal that are not acceptable. This message does not terminate the negotiation process. The following is the structure:

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

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

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

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

The design of the EBNM for Modify Proposal is patterned after three common modification operations: Update, Insert, and Delete. There are three levels of modification: attribute level, inter-attribute level, and inter-item level. Update and Insert have the same structure, but Delete has only one field. The reason is that it is enough to specify the data item name to be deleted. ATTR_UPDATE and ATTR_INSERT each have an additional field called ATTRTYPE. The assumption is that it is possible to change the attribute type, such as from an integer to a string.

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

```
WITHDRAW_PROPOSAL (HEADER, PID)
```

The EBNM for Acknowledge Message tells the negotiation partner that a message has been received. This message is optional for most negotiation primitives except the CallFor and Propose messages. The acknowledgement for CallFor and Propose is used to deal with two synchronization problems addressed in Section 4.3. This EBNM has only a header data structure, as shown below:

```
ACKNOWLEDGE_MESSAGE (HEADER)
```

8.1.3 Implementation of E-business Negotiation Messages Based on OAGIS

All the EBNMs described above are implemented following the structure and format of XML BODs proposed by OAG. There is a DTD for each negotiation message. HEADER is implemented using a common data type called NEGBODHEADER. Action names such as Propose, Reject, etc., are specified as the VERBs of BODs, and the data item names are mapped to the NOUNs. Structures such as NEGITEM and PROP are mapped to corresponding data types.

8.2 Component Description

8.2.1 Extensions to the Existing Negotiation Session Processor

The existing negotiation session processor uses ETR rules to relax the constraints and to determine whether a proposal should be rejected or not; however, other types of decision-making are hard-coded into the program. We have converted the portion of the program that governs the decision-making process into strategic rules stored in the rule base of the ETR server. During the runtime, when there is a need to make a decision about whether to accept the proposal, terminate a negotiation transaction, or generate a counterproposal, a synchronous event is posted to the ETR server. The results from the ETR server are used to generate the appropriate negotiation messages.

When a negotiation proposal/counterproposal is received, the negotiation system calls the CSP to check whether attribute constraints and inter-attribute constraints are satisfied or not. During the constraint satisfaction processing, CSP marks the violated constraints, if they exist. If all the constraints (both attribute constraints and inter-attribute constraints) are satisfied, the cost benefit evaluation module is called to pick the best alternative and send it to the counterpart. If there are violated attribute constraints, a synchronous event is posted for each violated constraint to test whether the violated attribute constraint is acceptable. The reason is that the acceptance condition may specify that the difference between the attribute values is tolerable (i.e., acceptable) if the difference is below a specified threshold. The acceptance testing may change some attributes from "unsatisfied" to "satisfied." After the acceptance testing, it is possible that multiple value combinations, which satisfied all the constraints (both attribute constraints and inter-attribute constraints), remain. In that case, the cost benefit evaluation module is called to pick the best alternative and send it to the counterpart. If all the violated constraints are inter-attribute constraints, a rejection message with a reason is sent to the counterpart, asking the counterpart to do some concession. Otherwise, it goes to the next step to do the termination testing.

In the termination step, a synchronous event is posted to the ETR server to determine whether the transaction should be terminated or not. If so, a termination message is sent to the counterpart. Otherwise, it goes to the next step for the counterproposal generation.

In the counterproposal generation step, concession functions for each violated attribute are applied. The result from the counterproposal generation step is sent to the counterpart as a counterproposal. One important difference between the approach taken in this dissertation and the one in the previous implementation is that here the concessions for all the attributes are done all together before a counterproposal is generated. Whereas, the previous approach performs concessions one attribute at a time and generates a counterproposal for each.

The existing negotiation session processor allows human interventions in the case of proposal modification and withdrawal. As pointed out in Chapter 5, it is possible to have rules to automatically modify or withdraw the previous proposals. ETR rules can be defined for this purpose. However, it is difficult to demonstrate this approach and functionality unless facilities are provided to simulate the condition for the modification or withdrawal of a previous proposal. In our work, ETR rules are added to modify the content of a previous negotiation proposal if the inventory drops to a certain level, and to withdraw a previous proposal if the current date is approaching the monthly or quarterly boundary. In a simulated inventory system, the inventory level drops a certain percentage after each message exchange. A rule is triggered if the user-defined threshold is reached. An artificial calendar is used to simulate the change in the date. Each time a message is received and a reply is sent, the calendar is increased by one day.

8.2.2 Knowledge Acquisition Processor

The GUI tools developed for a Knowledge Profile Manager (KPM) is 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 are 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 demonstrations of the negotiation server, is used for this purpose.

8.2.3 Knowledge Manager and Knowledge Repository

The Knowledge Manager is a program for managing the information stored in the Knowledge Repository. The Knowledge Repository is implemented using the Persistent Object Management (POM) [SHE01, WAN99] and the Metadata Manager being developed at the Database Systems Research and Development Center. Data retrieval and manipulation requests issued by the Knowledge Manager to the Knowledge Repository are posted in the SQL-like query language supported by POM. The Knowledge Manager provides a set of APIs to the Negotiation Plan Selector and the Knowledge Acquisition Processor to access and manipulate its contents.

8.2.4 Negotiation Plan Selector

The Negotiation Plan Selector 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 and action. The set of decision-action rules that implement a negotiation plan can then be used to drive the execution of the negotiation server.

8.2.5 ETR Server and Example ETR Rules

We have defined different types of decision-action rules in Chapter 5. In our implementation, the ETR server [LAM98, LEE00] developed at DSRDC is used to process these rules. When the negotiation server (see Figure 7.3) 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 counterproposal may involve more complicated computations. The main task in a counterproposal generation is to calculate concessions for various attributes. Some rules are shown and described below. The following is a part of a policy rule:

```
CONDITION [rulePar.roleName eq "S1"]

rulePar.mWorld.compInfo.companySize eq "F500" AND
rulePar.mWorld.compInfo.creditRating eq "excellent" AND
rulePar.mWorld.orderSize eq "large"

ACTION
     rulePar = new CommonData.RuleResponse();
     rulePar.initialResult.myGoal.p = 0.3;
     rulePar.initialResult.myGoal.t = 0.9;
     rulePar.initialResult.myGoal.d = 0.9;
     return rulePar;

ALTACTION
     rulePar = new CommonData.RuleResponse();
     return rulePar;
```

In order to simplify the description, related event and trigger information is not included. The above policy rule describes one possible corporate negotiation policy: If the counterpart is a medium-size company, and its crediting rating is good, set the P (Profitability) to 0.3, T (Time-to-agreement) to 0.9, and D (Desire-for-relationship) to 0.9. The following is a part of a strategic rule:

```
CONDITION [rulePar.roleName eq "S1"]

     (rulePar.finalResult.myGoal.p <= 0.4) AND
     (rulePar.finalResult.myGoal.t >= 0.6) AND
     (rulePar.finalResult.myGoal.d >= 0.8)

ACTION
rulePar = new CommonData.RuleResponse();
     rulePar.acceptancePrice = "S1A1Price";
     rulePar.acceptanceDelivery_period =
     "S1A1Delivery_period";
     rulePar.difPrice = 0.2;
     rulePar.difDelivery_period = 0.2;
     rulePar.terminateRuleName = "S1T1";
     rulePar.counterRuleName = "S1C1";
     rulePar.modifyRuleName = "S1M1";
     return rulePar;

ALTACTION
     return null;
```

The above strategic rule specifies that if P value is less than or equal to 0.4, T value is greater than or equal to 0.6, and D value is greater than or equal to 0.8, the following rules are chosen. Rule “S1A1Price” is chosen for the acceptance criterion of price, and Rule “S1A1Delivery_period” is chosen for the acceptance criterion of delivery period. The acceptable difference in price is 20%, and the acceptable difference in delivery_period is also 20%. The rule for termination is “S1T1”, the rule for counterproposal generation is “S1C1,” and the rule for modification is “S1M1.” The following is a part of a termination rule:

```
     CONDITION [rulePar.terminateRuleName eq "S1T1"]
     rulePar.numOfProposals > rulePar.terminateProposalNum
ACTION
     rulePar = new CommonData.RuleResponse();
     rulePar.terminateFlag = true;
     return rulePar;
ALTACTION
     rulePar = new CommonData.RuleResponse();
     rulePar.terminateFlag = false;
     return rulePar;
```

The above termination rule specifies that if the number of exchanged proposals exceeds a certain threshold specified by rulePar.terminateProposalNum, the negotiation transaction should be terminated.

8.3 Model Evaluation

In order to verify the validity of our decision model, we have conducted interviews and have studied “terms and conditions of sale” posted on some corporate websites. We have interviewed three people: an Associate Director of the purchasing division of a large educational organization, a small business owner, and a professor who teaches graduate negotiation courses. The general feedback is that the model is a new and original approach to the decision-making in business negotiations. The professor thought that our approach was “fascinating,” and the Associate Director thought our approach was imaginative and “out-of-the-box.” The small business owner thought our approach was innovative and feasible, but he pointed out that the approach was more suitable for large business organizations. The reason is that the distinction between the roles of the Policy Maker and the Negotiation Expert is clearer in large organizations than the distinction between those roles in small businesses.

The Associate Director, who has extensive experiences in both the private sector and the public sector, thought that systems incorporating our decision model might take a long time to be widely accepted by purchasing managers and sales managers. For example, his division still relies on conventional methods, such as telephone, fax, and email, to purchase goods/services. However, he pointed out that recently there was a change in the purchasing regulations in his organization. Previously, competitive bidding, RFQ (Request For Quote), and RFP (Request For Proposal) were the only ways to select vendors. Right now, his division has initiated a new process called Invitation to Negotiate (ITN). ITN gives purchasing agents the flexibility in deciding terms and conditions of business transactions. In our opinion, our decision model is useful for building negotiation systems to automate the ITN process in the future.

The interviewees have different opinion about the number of goal dimensions and their naming. All three interviewees agreed that our goal dimensions cover the important ones because time, money, and relationships are three important factors for policy makers. Other possible goal dimensions were discussed, such as quality of service (QoS) and timely shipment. These, however, could be covered by dimensions like relationship and time. The professor suggested that P (Profitability) be defined from a supplier's perspective. Therefore P (Profitability) should be changed to C (Cost) if the goal dimensions are defined from a buyer's perspective. Three goal dimensions are not completely independent. For example, if company A wants to have a good relationship with company B, company A may need to reduce the value of the profitability dimension.

Corporate websites usually post documents called "terms and conditions of sale." We have studied a number of corporate websites, especially websites of some computer companies. Although most terms and conditions are lists of non-negotiable items, like applicable laws and taxes, there are some negotiable items on some websites. For example, terms of payment can be either 30 days or 45 days. The warranty can be 90 days, 1 year, 2 years, or 3 years. One website lists three dispute resolution methods: face-to-face negotiation, mediation, and binding arbitration. In our opinion, it is possible to have another goal dimension called “customer satisfaction” that is linked to issues like terms of payment, warranty period, and conflict resolution methods.

CHAPTER 9
CONCLUSIONS AND FUTURE WORK

9.1 Conclusions

In this dissertation, we first gave the motivation for conducting the research into automated e-business negotiation. We then pointed out that, in spite of many efforts by researchers of various disciplines to investigate different aspects of e-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 dissertation, we have defined such a model. We have addressed two important aspects of the negotiation model: a decision model and negotiation messages. The decision model formally defines business enterprises' negotiation contexts, goals, policies, plans of decision and action, strategies, and their inter-relationships. We have proposed a quantitative way to define negotiation goals and, thus, allow more flexibility in mapping goals to plans of decisions and actions by strategic rule specifications. We have also proposed to use parameterized decision-action rules to specify and implement plans of decisions or actions.

We have defined a set of negotiation messages for conducting e-business negotiations. These messages are based on the proposed negotiation primitives implemented in the existing negotiation server. Two negotiation scenarios are used to show the exchange of negotiation messages. The synchronization problems associated with Modify Proposal and Withdraw Proposal are discussed.

In this work, we also combined and extended the negotiation life cycle model proposed by Robinson and Volkov [ROB98] 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 and 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 better match with the different roles that people involved in negotiations play, but also because they provide more flexibility for making changes to negotiation contexts 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 dissertation, we have 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 existing negotiation server. It shows how the additional components for implementing the decision model interact with the implemented components of the existing negotiation server. A prototypical implementation of the new system architecture has also been described in this dissertation.

The decision model, the life cycle model, and the negotiation messages are three major contributions of this dissertation. These contributions have become the newly added building blocks to construct a new system architecture of an automated negotiation system. The decision model introduces new concepts such as negotiation context, negotiation goal and negotiation plan, plus mappings to link them. Although previous work occasionally addresses concepts such as goal, policy, and strategy, but none of them linked these concepts together and studied them in the context of e-business negotiation. We have conducted three interviews to verify the validity of our decision model. The negotiation messages introduced in this work are a comprehensive set of messages for bargaining type of negotiations. From our survey, we conclude that our negotiation messages are a superset of messages proposed in previous work. Our research on the decision model and negotiation messages resulted in the enhancement of the functionality of the existing negotiation system, which deals with isolated negotiation transactions. The life cycle model adds an additional dimension: time. Our four-phased life cycle model divides a negotiation process into four phases, making each phase more manageable. The model also has a feedback mechanism to let the current negotiation outcome influence future negotiation transactions. In our opinion, the four-phased life cycle model closely matches human-based business negotiations.

We feel that more work needs to be done in the area of dynamic concession, discussed in Subsection 5.3.7.2. Concession is one of the cornerstones of business negotiations. The failure or success of a negotiation largely depends on the speed and the timing of mutual concessions. Although we have proposed two methods for dynamic concessions, we have not tested the validity of the assumptions. In the first method, we assume that the attribute under negotiation is independent of the other attributes. In order to reduce the complexity, it is possible (even desirable in some cases) to negotiate attributes separately. However, attributes are often negotiated together in human-based negotiations. In the second method, we assume that the counterpart's importance factor is inversely proportional to the counterpart's concession speed for that attribute. Generally speaking, the assumption is valid. For example, if the counterpart were very sensitive to price, the counterpart in general would make little concession on price. Therefore, it makes sense for our own negotiation system to assign a relatively large value to the importance factor for price. However, if the counterpart tries to "outsmart" our system by guessing and manipulating the concession algorithm used by our system, our assumption about the counterpart would be questionable. The situation gets more complicated if our own negotiation system wants to "outsmart" the counterpart that wants to "outsmart" ours. In this dissertation, we did not consider complicated situations like these.

9.2 Future Work

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

9.2.1 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 2-D diagram to represent the utilities of both parties, the solutions are points in the 2-D space. Figure 9.1 shows the two dimensions. Some solutions are shown as points and the Pareto frontier is shown as 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, cannot be further improved (i.e., cannot be moved further toward the up-and-right direction). A solution, S1, is said to dominate another solution, S2, if S1 improves the utility for both parties, or improves the utility for one party but does not harm that of the other party. It is clear from Figure 9.1 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 disadvantages to one of the parties.

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.

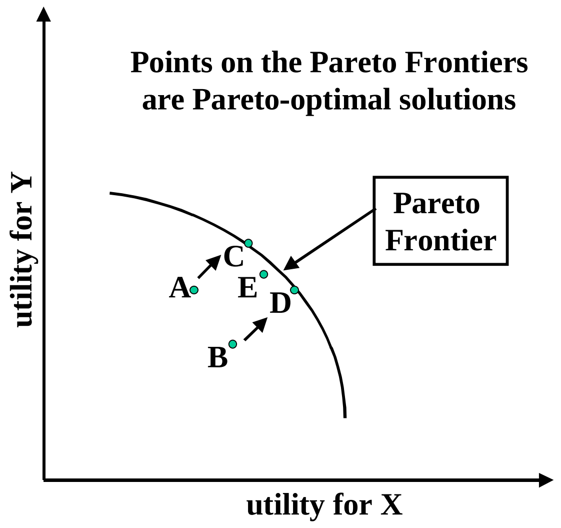

Figure 9.1 Pareto-optimal Solutions

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 actually 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.

9.2.2 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 dissertation 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 bilateral negotiations. However, due to the added complexity, many additional problems need to be dealt with. The progress and the result of each bilateral negotiation need to be coordinated with those of the other bilateral negotiations. The decision to be made in one thread of the negotiation may depend on the progress or result of another. 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 Best Alternative To the Negotiated Agreement (BANTA): 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, Third Edition, Houghton Mifflin Company, Boston, MA, 1996.

[BAR95] M. Barbuceanu and M. S. Fox, "COOL: A Language for Describing Coordination in Multi-Agent Systems," Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, CA, pp. 17-24, 1995.

[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, UK, September 2000.

[BEN98] I. Ben-shaul, Y. Gidron, and O. Holder, "A Negotiation Model for Dynamic Composition of Distributed Applications," 9th International Workshop on Database and Expert Systems Applications (DEXA'98), Vienna, Austria, August 1998.

[BIN99] K. Binmore and N. Vulkan, "Applying Game Theory to Automated Negotiation," Netnomics, Vol. 1, No. 1, pp. 1-9, 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, London, UK, 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, No. 3, pp. 97-125, Winter 1996.

[FEL99] S. Feldman, Keynote speech at ACM 1999 Conference on OOPLSA, URL: http://www.ibm.com/iac/oopsla99-sifkeynote.pdf, 1999.

[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, Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL, 2000.

[JEN00] N. R. Jennings, S. Parsons, C. Sierra, and P. Fartin, “Automated Negotiation,” Proceedings of 5th International Conference on the Practical Application of Intelligent Agents and Multi-Agent Systems (PAAM-2000), Manchester, UK, 2000.

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

[KAN98] J. Kang and E. Lee, “A Negotiation Model in Electronic Commerce to Reflect Multiple Transaction Factors and Learning,” 13th International Conference on Information Networking (ICOIN '98), Tokyo, Japan, January 1998.

[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, pp. 135-154, 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, pp. 1-2, 1998.

[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, pp. 47-53, 1998.

[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, Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL, 2000.

[LI01] H. Li, C. Huang, and S. Y. W. Su, “Design and Implementation of Business Objects for Automated Business Negotiations,” accepted by Group Decision and Negotiation, 2001.

[LIM93] L. Lim and I. Benbasat, “A Theoretical Perspective of Negotiation Support Systems,” Journal of Management Information Systems, Vol. 9, No. 3, pp. 27-44, 1998.

[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, September 2000.

[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, pp. 81-91, March 1999.

[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.

[MAR98] K. Marriott, and P. J. Stuckey, Programming with Constraints: An Introduction, MIT Press, Cambridge MA, 1998.

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

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

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

[MOR00] Morgan Stanley Dean Witter, “The B2B Internet Report – Collaborative Commerce,” URL: www.msdw.com/techresearch/index.html, 2000.

[MÜL96] H. J. Müller, “Negotiation Principles,” Chapter 7, Foundations of Distributed Artificial Intelligence, G. M. P. O’Hare and N. R. Jennings, eds., John Wiley & Sons, New York, NY, 1996.

[MYE83] R. Myerson and M. Satterthwaite, “Efficient Mechanisms for Bilateral Trading,” Journal of Economic Theory, Vol. 29, pp. 265-281, 1983

[NIS97] M. Nissen, “The Commerce Model for Electronic Redesign,” Journal of Internet Purchasing, URL: 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, pp. 83-112, 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, pp. 1147-1163, 1997.

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

[ROU99] Routing Policy Specification Language, URL: 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/proceedings/00jul/I-D/ipsp-spsl-00.txt, 2000

[SEG98] A. Segev and C. Beam, New Market-based Negotiation Paradigm, URL: http://www.haas.berkeley.edu/citm/nego_proj/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.

[SHE01] A. Shenoy, "A Persistent Object Manager for Java Applications," Master's Thesis, Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL, 2001.

[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, pp. 17-35, 1997.

[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, pp. 1104-1113, 1980.

[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, pp. 472-520, September 1987.

[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, January 4-7, 2000.

[SU00b] S. Y. W. Su, "EECOMS's Negotiation Primitives and Negotiation Protocol," EECOMS internal document, 2000.

[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, pp. 294-296, 1985.

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

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

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

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

[TSA93] E. Tsang, Foundation of Constraint Satisfaction, Academic Press, New York, NY, 1993.

[UNI00] University of Maryland, Baltimore County, “Negotiation Rules Learning and Related Issue,” EECOMS internal document, 2000.

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

[WAN99] L. Wang, “A Persistent Object Manager and Query Language Interface for Object-oriented Applications,” Master’s Thesis, Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL, 1999.

[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, pp. 125-141, 1998.

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

BIOGRAPHICAL SKETCH

Haifei Li was born on March 31, 1969, in Xinhua County, Hunan Province, China. He graduated from Xi'an Jiaotong University in 1990, with a Bachelor of Engineering in computer science and engineering. He worked for the Geophysical Research Institute (GRI) of the China National Petroleum Corporation (CNPC) after graduation. He participated in the development of two large scale systems: GRISYS/90, a seismic data processing system, and GRIstation, an integrated seismic and geologic interpretation system during his stay at GRI from 1990 to 1994. From 1994 to 1996, he was a software/system engineer for China Resources Information Technology, Inc., (CRIT).

From 1996 to 1998, he was a student member of the Database Systems Research and Development Center, and a master's student in the Department of Electrical and Computer Engineering, University of Florida. He got his master's degree in August 1998, and then joined the Department of Computer and Information Science and Engineering as a doctoral student in computer science. His research interests include e-business, B2B integration, and databases.