Université de Montréal

Combined Negotiations in E-Commerce: Concepts, Architecture, and Implementation

Par

Morad Benyoucef

Département d'Informatique et de Recherche Opérationnelle

Faculté des Arts et des Sciences

Thèse présentée à la Faculté des études supérieures en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.) en Informatique

Juillet, 2002

Université de Montréal

Faculté des études supérieures

Cette thèse intitulée :

Combined Negotiations in E-Commerce : Concepts, Architecture, and Implementation

Présentée par :

Morad Benyoucef

a été évaluée par un jury composé des personnes suivantes :

Rudolf K. Keller	Directeur de recherche
Peter Kropf	Président - rapporteur
Julie Vachon	Membre du Jury
Gregory Kersten	Examinateur externe
Gilbert Babin	Représentant du doyen de la FES

Thèse acceptée le 27 août 2002

SOMMAIRE

L'importance des négociations dans les marchés traditionnels ainsi que dans les marchés électroniques a longtemps été reconnue. Aussi, tout le monde reconnaît le fait que les négociations humaines ont tendance à être complexes, et consomment du temps et des ressources. Un support logiciel à ce genre de processus s'avère donc nécessaire. De nos jours, cette nécessité est entrain de changer en défi, vu que de nouveaux protocoles de négociation sont introduits dans les marchés électroniques. Les Négociations Combinées sont un exemple de tels protocoles.

Dans une Négociation Combinée (CN), l'usager, qu'il soit un individu ou une entreprise, est intéressé par un ensemble d'items (biens ou services), et par conséquent, s'engage dans des négociations pour tous ces items en même temps. Les négociations sont indépendantes les unes des autres, alors que les items sont typiquement interdépendants. Utiliser les technologies de support aux négociations électroniques revient, pour l'usager, à conduire chaque négociation séparément, et à coordonner et réconcilier les différentes négociations.

Dans cette recherche doctorale nous introduisons et définissons les CNs, identifions les questions de recherche qu'elles engendrent, et proposons, concevons, implantons et validons une solution logicielle pour les conduire. La solution prend la forme d'un système de support aux négociations combinées (CNSS) qu'un usager utilise pour : (1) trouver les items et les fournisseurs de ces items; (2) construire un modèle de CN qui reflète le cheminement des négociations individuelles et les dépendances entre elles; (3) fournir les stratégies de négociation et les canevas de coordination; et (4) exécuter le modèle en intervenant quand c'est nécessaire.

Nous avons conçu un CNSS que nous appelons CONSENSUS, basé sur l'idée suivante: (1) un workflow qui capte le cheminement de la CN; (2) des agents logiciels qui conduisent les négociations individuelles; (3) des règles si-alors pour doter les agents de stratégies de négociation; et (4) des règles si-alors pour gérer la coordination entre agents. La technologie des workflows apporte une grande fiabilité, les agents logiciels apportent l'automatisme, et les règles si-alors apportent la modularité, l'uniformité, un niveau élevé d'abstraction, une versatilité et une facilité d'utilisation, ainsi que la possibilité de changer les connaissances dynamiquement. Un défi important était de répartir les fonctionnalités entre ces quatre composants (workflow, agents, stratégies, et coordination) d'une façon propre et optimale.

Nous avons implanté CONSENSUS et nous l'avons utilisé pour modéliser et exécuter plusieurs cas de CN simples et complexes. Pour cela, nous avons utilisé GNP, notre serveur de négociation, pour implanter, déployer et exécuter plusieurs types de négociations, et nous les avons utilisées comme nœuds dans des CNs. Les résultats sont satisfaisants. Nous avons utilisé le même décor pour conduire des tournois de mises dans le but de montrer qu'une représentation déclarative peut être utilisée pour doter les agents de simples tactiques de mises et de coordination. Les résultats sont encourageants, et le même décor d'expérimentation peut être utilisé dans une recherche sur le mécanisme de représentation lui-même ou pour l'évaluation de stratégies de négociation connues ou nouvellement conçues.

Mots clés : négociation, enchère, commerce électronique, négociation électronique, négociation combinée, workflow, agent logiciel, stratégie de négociation, moteur de règle, coordination.

ABSTRACT

The importance of negotiations in traditional and electronic markets has long been recognized by the academic and business communities. Also recognized is the fact that human negotiations tend to be complex, and time and resource consuming. Therefore, software support in conducting such processes is necessary. Nowadays, this necessity is becoming a challenge, as new negotiation protocols are introduced within electronic markets. One such protocol is Combined Negotiations.

In a Combined Negotiation (CN), the user, be it a consumer or a company, is interested in a package of items (goods or services), and consequently engages in negotiations for all the items at the same time. The negotiations are independent of each other, whereas the items are typically interdependent. Using currently available e-negotiation support technology, the user would have to conduct each negotiation separately, and would have the burden of coordinating and reconciling them.

In this doctoral research we introduce and define CNs; point to their challenges; identify the research issues they raise; and suggest, design, implement, and validate a software solution to conduct them. The solution takes the form of a Combined Negotiation Support System (CNSS) that a human can use to: (1) find the items and the providers of these items; (2) build a CN model that captures the sequencing of the individual negotiations and the dependencies between them; (3) provide negotiation strategies and coordination schemes; and (4) run the model, and possibly intervene in it when necessary.

We designed an architecture for a CNSS we call CONSENSUS. The driving idea behind it is: (1) a workflow that captures the sequencing and the control flow of the

CN; (2) software agents that carry out the individual negotiations; (3) if-then rules to provide the agents with negotiation strategies; and (4) if-then rules to manage agent coordination across several negotiations. Workflow technology brings a high level of reliability; software agents bring automation; and if-then rules provide modularity, uniformity, a high level of abstraction, versatility, ease of use, and the possibility to modify the knowledge dynamically. An important challenge was to allocate functionalities to the four components (workflow, agents, strategies, and coordination) in a clean an optimal way.

We implemented CONSENSUS and used it to model and enact several simple and complex CN cases. We used our negotiation server GNP to implement, deploy, and run several negotiation types, and used them as CN nodes. The results were satisfying. We used the same setting to conduct bidding tournaments to show that a declarative representation can be used to provide software agents with simple bidding tactics and coordination schemes. The results were encouraging, and the experimentation setting can be used for further investigations regarding the representation mechanism itself, or for the evaluation of known and newly designed negotiation strategies.

Keywords: negotiation, auction, electronic commerce, electronic negotiation, combined negotiation, workflow, software agent, negotiation strategy, rule engine, coordination.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI: Artificial Intelligence
ARCHON: Architecture for Cooperative Heterogeneous ON-line systems
B2B: Business to Business
B2C: Business to Consumer
CBB: Consumer Buying Behavior
CBR: Case Based Reasoning
C2C: Consumer to Consumer
CDA: Continuous Double Auction
CN: Combined Negotiation
CNSS: Combined Negotiation Support System
DOM: Document Object Model
DSS: Decision Support System
EJB: Enterprise JavaBeans
ETR: Event-Trigger-Rule
FCC: Federal Communication Commission
FSM: Finite State Machine
JSP: Java Server Pages
KIF: Knowledge Interchange Format
MAS: Multi Agent System
NSS: Negotiation Support System
GEE: Generic Experimentation Engine
GNP: Generic negotiation Platform
OMG: Object Management Group
PCS: Personal Communication Services
RDBMS: Relational Data Base Management System
RFP: Request For Proposal
SOAP: Simple Object Access Protocol
TEM: Towards Electronic Marketplaces
UDDI: Universal Description, Discovery and Integration
UML: Unified Modeling Language
WfMC: Workflow Management Coalition
WfMS: Workflow Management System
WSDL: Web Services Description Language
WWW: World Wide Web
XMI: XML Metadata Interchange
XML: eXtensible Markup Language

THANKS

I would like to thank my supervisor Rudolf "Ruedi" Keller, professor at the University of Montreal, for giving me the chance to join his team, and for his trust and generosity. His unique approach, sense of organization, and genuine optimism were a determinant factor in the success of this work.

My thanks also go to Peter Kropf, professor at the University of Montreal, for presiding my jury and for his constructive remarks. Julie Vachon, professor at the University of Montreal and member of my jury, may also find here the expression of my appreciation. Furthermore, I would like to thank Gilbert Babin, professor at HEC, for representing the Faculty Dean at my thesis defense.

My appreciation also goes to Gregory Kersten, professor at Concordia University and leader of the InterNeg Group for accepting to be the external examiner. His contribution to e-negotiations and his leading role in Negotiation Support Systems research were a reference and an inspiration to me throughout this thesis project.

I would also like to thank CIRANO and Bell Canada for supporting me financially, and for making my PhD years an enjoyable experience. My special thanks go to Jacques Robert, professor at HEC and scientific director of the TEM project, for his leadership, and to Robert Gerin-Lajoie for his help and the valuable remarks he made during our numerous discussions. Thanks to Alan Bernardi, Director of Bell's University Laboratories for his help. All CIRANO's researchers and employees should find here the expression of my sincere appreciation, particularly: Eveline Dufort, Sophie Lamouroux, Kim Levy, Frederic Loisier, Vincent Trussart, and Mathieu Vezeau.

Thanks to my colleague and friend Sarita Bassil for her contribution to this project and for assuring the further evolution of the CONSENSUS project.

Thanks to my colleague and friend Hakim Alj for his technical and moral support.

Thanks to my family for understanding my desire to go back to my studies, and for their endless love and support. Thanks to Halima for her patience, and for sharing the stress of this thesis with me.

Morad

Hawthorne, New York, June 2002

To the memory of my mother

To the memory of Ahmed, my brother

"... always aiming high, always standing."

CHAPTER 1

INTRODUCTION

1.1 Background

What is commerce? According to the Object Management Group[1] (OMG), commerce is at its heart an exchange of information. To obtain something of value from someone else, you need to: (1) find a person who has what you want and communicate your desire, (2) negotiate the terms of a deal, and (3) carry out the deal. Each one of these activities involves an exchange of information [ECD97]. Negotiation is defined as mechanisms that allow a recursive interaction between a principal and a respondent in the resolution of a good deal [OMG99]. The principal and the respondent are usually a consumer and a supplier. In general, they need to negotiate the price, the delivery date, the conditions of the purchase, the terms of the guarantee, etc.

Nowadays, commerce is shifting to the Internet, and a dramatic increase in the number of companies doing business on the Web has occurred. This has led to the success of electronic commerce (e-commerce). E-commerce can be thought of as a series of increasingly general scenarios for business exchange via, or supported by, modern computers and communication technology [Oli97]. A simple form of e-commerce is the creation of a site on the World Wide Web (WWW) describing the goods and services offered by a company, and a simple e-commerce transaction can take the following form: the company receives purchase orders from consumers (individuals or other companies), delivers the goods or services, and accepts the payments. Major steps of the transaction, if not all, are carried out electronically. This simple paradigm can be taken a step further by allowing the consumer to negotiate

the price, the delivery date, the conditions of the purchase, the terms of the guarantee, etc. If the negotiation is carried out electronically, then we talk of an electronic negotiation (e-negotiation).

Negotiation is considered a key component of e-commerce [San99a]. The MIT Media laboratory[2] (through the AMEC[3] project, conducted by Pattie Maes), for instance, recognizes its importance by including it in its Consumer Buying Behavior (CBB) model for e-commerce. The model identifies six steps in an e-commerce transaction [MGM99]: (1) the need identification step in which the buyer is stimulated through product information, (2) the product brokering step in which information is retrieved to help the consumer determine what to buy, (3) the merchant brokering step in which the consumer determines who to buy from, (4) the negotiation step where the price and possibly other aspects of the deal are settled, (5) the purchase and delivery step, and finally (6) the product and service evaluation step where the product and service are evaluated by the consumer.

The business community was also fast to recognize the importance that negotiation should play in conducting business over the Internet[4]. "Negotiated Trade: The Next Frontier for B2B e-commerce", a June 2000 report by the Hurwitz Group[5], claims that 80% of commerce is performed through negotiated trade, and defines a typical B2B transaction as a four-step process [Hur00]:

(1) Search: the buyer company publishes a Request For Proposal (RFP) to multiple suppliers, and one or more suppliers respond to the RFP.

(2) Evaluation and negotiation: the buyer company reviews the proposals, and both parties hash out the terms of the proposals through negotiations. Negotiation variables may include price, availability, payment, shipment terms, etc.

---

[1] http://www.omg.org
[2] http://www.media.mit.edu/
[3] Agent Mediated E-Commerce, one of the pioneering projects to address negotiations in e-commerce.
[4] Although the business community tends to focus on B2B commerce, we will see that B2C and C2C are fertile ground for negotiation research as well.
[5] http://www.hurwitz.com

(3) Contract[6] generation: a successful negotiation leads to a contract that documents the terms and conditions of the deal.

(4) Contract management: this is sometimes also called fulfillment or execution, and it consists of carrying out the deal while monitoring every action closely to make sure it conforms to the terms of the contract.

The following brief look at e-negotiations is necessary to set the stage for presenting the problem statement.

The most basic form of e-negotiation, as described by Kumar et al. [KF98b], is no negotiation at all (also called fixed-price sale) where the seller offers goods or services through a catalogue at take-it-or-leave-it prices. This is the way most electronic retailers (e-tailers) such as Amazon[7] operate. Auctions are a bit more complex, and they are at present the most visible type of e-negotiations on the Internet as conducted by eBay[8], Yahoo[9] and hundreds of other auction sites. Auctions are preferred to fixed-price sales in domains such as fine arts, perishable goods, etc. The reason for that is that these goods are of uncertain value, and dynamic price adjustment will often maximize revenue for the seller [PUF99]. Moreover, on-line auctions (i.e., e-auctions) can reach a large and physically distributed audience at reduced cost, whereas off-line auctions tend to cost more, and require that the participants gather in one physical location. E-negotiations can take an even more complex form called bargaining (i.e., haggling). This involves making proposals and counter-proposals until an agreement is reached [San99b]. Bargaining can be bilateral or multi-lateral, depending on whether there are two parties (one-to-one bargaining) or many parties (many-to-many bargaining) involved in the negotiation [OMG99]. If the object of the negotiation has more than one negotiable attribute (e.g., the price, the quality, and the delivery date), then we talk of multi–attribute negotiations. Multi-attribute negotiations are already supported by e-commerce software solutions such

---

[6] A contract is usually the result of a B2B negotiation, but even the simplest B2C online purchase transaction involves an implicit contract.
[7] http://www.amazon.com
[8] http://www.ebay.com

as the Negotiated Commerce Engine by MOAI[10]. Finally, combinatorial auctions [San99b, SL95] involve making bids on combinations of items, and one of the main challenges is for the auctioneer to determine the winning bid. Several solutions to this problem have been proposed, one of which is *iBundle* by David Parkes [Par99].

This thesis is part of a considerable effort by the research community dedicated to the subject of negotiations in general and e-negotiations in particular. An increase in the number of conferences and workshops on this subject has occurred, attracting competences from horizons as diverse as software engineering, economics, decision science, management science, and operations research. One of the most recent workshops on the subject is: *"The Third International DEXA[11] Workshop on Negotiations in Electronic Markets – beyond price discovery, Aix en Provence, France, September 2-6, 2002"*.

We would like to contribute to the rich and challenging e-negotiations field by introducing a new negotiation type we call *Combined Negotiations*. Simply stated, a Combined Negotiation takes place when a consumer is interested in a package of interrelated items (goods or services), and in general engages in many independent negotiations with the providers of the items. The negotiations can be of any type (fixed-price sale, English auction, Dutch auction, bilateral bargaining, combinatorial auction, etc.). The negotiation process being generally long and mostly manual, software support for all or some of its aspects will provide tangible benefits [Hur00]. We would like to provide *software support* to the human negotiator in conducting Combined Negotiations, and *automate* certain routine and tedious negotiation tasks.

This doctoral research was conducted as part of the TEM (Towards Electronic Marketplaces) project, a joint industry-university project started in 1999 involving researchers from economic science, software engineering, and operations research. TEM addresses market design issues in respect to resource allocation and control and reward mechanisms, investigates open protocols for electronic marketplaces, and

---

[9] http://www.yahoo.com
[10] http://www.moai.com

explores concepts and tools for e-negotiations, based on the belief that e-negotiations may be re-engineered from traditional negotiations, or designed expressly for electronic marketplaces.

This research was supported by Bell[12] Canada through its Bell University Laboratories R&D program, by NSERC[13] (Natural Sciences and Engineering Research Council of Canada, CRD-224950-99), and by CIRANO[14] (Centre Interuniversitaire de Recherche en ANalyse des Organizations).

1.2 Problem Statement

We begin by giving an example of a *Combined Negotiation* (CN). Let us suppose that a consumer is interested in a vacation package consisting of three items: a transportation ticket, a hotel room, and a ski trip. The three items are obviously interrelated since the consumer would have to travel to the location where the ski trip journey initiates (or at least near it) on the date of the trip (or before it). In addition to the places and dates, there can be other constraints and dependencies between the three individual items, such as the total amount to be spent on the vacation, the maximum price the consumer is willing to pay for the ski trip, the duration of the vacation, the preferences the user has for certain vacation destinations, her schedule constraints, etc. Let us also suppose that the three items are negotiable (keeping in mind that a fixed-price sale is a special case of a negotiation), that they can be negotiated on different negotiation servers (or on the same server, but run as separate negotiations), and that the individual negotiations are independent of each other (from the server's point of view, or the servers' point of view if more than one server is involved). We suppose also that the individual negotiations can be of different types ("type" in this context means "the rules of the negotiation"; synonymously, the terms "process", "protocol" and "style" are sometimes used).

---

[11] http://www.dexa.org
[12] http://www.bell.ca
[13] http://www.nserc.ca
[14] http://www.cirano.qc.ca

Where does this new type of negotiation (i.e., the CN) fit in the already crowded universe of negotiation types? To answer this question we rely on a three-dimensional negotiation space (see Figure 1.1, solid lines) suggested by Jhingran [Jhi99]. The first dimension of the 3D space is for the case where *multiple copies* of the same item are available for negotiation, and the bids (i.e., offers, counteroffers) take the form of pairs (quantity, price-per-unit). The second dimension addresses the case where *multiple items* are subject to one negotiation. In this case, the participants make bids on combinations of these items. The third dimension, finally, is for *multiple attribute* negotiations. We have a CN when a consumer engages in many negotiations for different items at the same time. The negotiations can be of any type (fixed-price sale, Dutch auction, English auction, bilateral bargaining, combinatorial auction, etc.). Each individual negotiation is for a separate bundle of copies, items, and attributes and thus corresponds to one point in the 3D negotiation space (see Figure 1.1, solid lines). The negotiations are in general totally independent of each other. The goods and services of the CN, however, are typically interdependent.

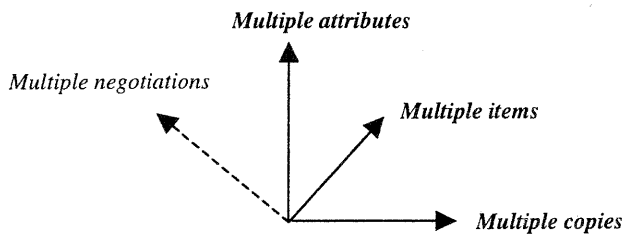

Figure 1.1: The four-dimensional negotiation space.

To capture this new type of negotiation, where we have multiple negotiations going on at the same time, we introduced a fourth dimension in Jhingran's model (see Figure 1.1, dashed line).

Using currently available negotiation technology, the consumer would have to conduct each negotiation separately, and she would have the burden of coordinating and reconciling the various negotiations. It can happen for instance that the consumer makes a deal on a plane ticket and a hotel room, and then, while negotiating the ski trip, finds out (through bargaining for example) that she is missing out on a very

interesting deal only because she arrives at the vacation site a few hours late. It can also happen that she finalizes deals on the hotel room and the ski trip but cannot get a good deal (or any deal at all) on the plane ticket. In most cases, the obvious thing to do in this latter case would be to back off and break the commitments she has already made. Unfortunately, this is not always permitted, and even if it is, it usually costs money and leaves the consumer at the point where she started (i.e., with no vacation package altogether).

We therefore see the need for a system to support the consumer in conducting all the negotiations at the same time, i.e., in carrying out CNs, and we call such a system a *Combined Negotiation Support System* (CNSS). A CNSS is a tool that enables the user to track and monitor the progress of many negotiations efficiently and to respect all the constraints, dependencies and preferences of the given context. Moreover, a CNSS will support the user in taking decisions.

Using a CNSS bears some similarity with modeling a software system, in that a high-level specification component will serve to specify the sequencing and the dependencies between the individual negotiations, and a human-readable syntax will be used to specify the negotiation tactics and strategies. Running the model (i.e., the enactment part), and interacting with it (i.e., the tracking part) will also be supported by appropriate software components.

The envisioned solution will take the form of a support tool, where the automation provided by the run-time system should allow for user intervention in the important aspects of the negotiation whenever a need should arise. A CNSS is not meant to replace the user, but to provide her with a powerful tool. To meet that important goal, we study and leverage Negotiation Support Systems (NSSs) [KNT02, KL01, KS98].

The envisioned support tool should also enable the user to discover the items and the providers of the items she is interested in (i.e., Step 2 and Step 3 of the CBB model – see Section 1.1), and the necessary information to help her decide which items to negotiate and where to negotiate them. The user should know, for instance, the negotiation type practiced by the corresponding servers, as such information is an

important factor in choosing one provider over another. She might for example prefer a bargaining type negotiation to an English auction if she is a good haggler. Therefore, special attention will be devoted to capturing negotiation protocols. We believe that these protocols are as important to the negotiating parties as the goods themselves or the tactics being employed in the process.

The vacation package scenario is a case of *Business to Consumer* (B2C) e-commerce. If some items in the package were offered by consumers (i.e., a rare ticket to a rock concert offered on an auction site), we would face a *Consumer to Consumer* (C2C) transaction. A CNSS can also be used at the *Business to Business* (B2B) level. A travel agency, for instance, can use a CNSS to negotiate travel packages on behalf of its clients. The more items there are to be negotiated and the more providers of such items there are, the more there is a need for a support tool. We might imagine different means of transportation (air, train, bus, etc.), different types of accommodation (hotel, motel, pension, etc.), and different recreational activities (skiing, camping, hiking, theater, etc.). We might also imagine that there are many providers of such services and that each provider might practice a different type of negotiation. As another example at the B2B level, consider a company that wants to import some merchandise from a foreign country. The attributes and issues to be negotiated include: the price as well as other attributes of the merchandise, the transportation, the insurance, the customs, etc. These examples suggest that at the B2B level, CNs may become highly complex, and that automated support by a CNSS is even more important.

1.3 Research Objectives

According to Kersten et al., the explosive growth in e-commerce has not reduced the complexity of negotiations conducted over the web, partly due to human factors, and partly because the underlying economic models remain unchanged, despite the increase in speed, reach, and computational efficiency [KL01]. The objectives of this research are twofold. First, we aim to contribute to reducing the complexity of negotiations by way of software support and automation. Second, we will define a

new negotiation model, identify the problems it generates, propose solutions to these problems at both the conceptual and the tool level, and evaluate these solutions. These objectives are developed further in the following points:

(1) Define precisely the Combined Negotiations model and show the need for a software tool to conduct it.

(2) Define a set of functional and non-functional requirements for the envisioned tool.

(3) Determine how much automation the tool should provide and how much user involvement will be necessary.

(4) Explore architectures for the envisioned tool and study their feasibility.

(5) Design a complete architecture for the envisioned tool with the following objectives:

   a. Pragmatic and realistic.

   b. Based on reuse, and off-the-shelf components.

   c. Modular, and resilient to evolving functionality.

   d. Flexible enough to be used as a research infrastructure.

(6) Implement prototypes as a proof-of-concept of the proposed architecture.

(7) Use the prototypes to validate the architectural choices.

(8) Since this research is part of the TEM project, we are able to use the tools developed in the context of the overall project. We rely, for instance, on our negotiation server GNP [BKL+00]. Our first objective regarding GNP was to contribute to its design by defining and refining some of the requirements such as its generic aspect, the ease to implement new negotiation protocols, and the API for connecting software agents.

1.4 Publications Trail

This research led to the publication of several papers.

The first one, written early on in the project, was dedicated to GNP. It was recognized from the very beginning of our doctoral research and of the TEM project alike, that a negotiation platform would be needed. The design of such a platform was therefore of great importance, and special effort and attention were dedicated to it, which resulted in GNP (Generic Negotiation Platform). The design process was documented in [BKL+00]. We present a modified version of this paper in Chapter 5.

Our study of the various negotiation protocols and the ways they are usually represented led to a second paper [BK00a]. This paper also introduced our own representation scheme, which relies on UML statecharts to describe negotiation protocols. A modified version of this paper is presented in Chapter 3.

The results of our initial efforts to tackle the issues surrounding Combined Negotiations were reported in [BK00b], a paper that was later developed into a more complete publication [BAV+01], detailing Combined Negotiation issues, and presenting a more mature architecture for the proposed support tool, as well as the implementation details. Chapter 4 is a modified version of this latter paper.

Ways to represent and manage negotiation strategies and coordination schemes were investigated as part of the global solution, resulting in two publications. The first one [BAK01] introduced the rule-based approach we adopted for our negotiating agents. The second paper [BAK02] was a refined version of the first one. It detailed and justified the solution, and presented the results of its validation. Chapter 6 is a modified version of this latter paper.

Workflow technology was also important in our proposed solution, thus two publications were dedicated to this topic. The modeling of Combined Negotiations, and the use of software agents as actors in workflows were described in [BBK01]. Dynamic workflow systems and their possible integration into our solution made the subject of [BBK+02].

Finally, a comparison of e-negotiation systems in light of the e-negotiations Group classification [NBB+02] has been submitted to publication in the Group Decision and Negotiation Journal.

1.5 Major Contributions

The major contributions of this thesis can be stated as follows:

(1) Definition of a new negotiation type (i.e., Combined Negotiations), and identification of the issues and difficulties it generates.

(2) Design of an architecture for a tool to support the user in tackling these issues and coping with these difficulties.

(3) Knowledge management and functional allocation in the support tool. The tool comprises four major parts: *a workflow* that captures the sequencing and the control flow of the CN (*CN know-how*), *software agents* that carry out individual negotiations (*individual negotiation know-how*), strategy *rules* to help the agents decide what to do when there are many options to choose from (*negotiation strategy know-how*), and *coordination rules* to manage the agents across several negotiations (*coordination know-how*). An important challenge in this research was to allocate functionalities to these four parts in an optimal way.

(4) Implementation and test of a proof-of-concept prototype of the proposed architecture.

(5) Validation of our architectural choices using GNP as well as off-the-shelf software tools such as commercial workflow systems and rule engines.

(6) Proposal of a UML based representation of negotiation protocols and application to the generic aspect of GNP.

(7) Adoption of workflow technology to model and enact Combined Negotiations.

(8) Monitoring of software agents by workflows. Workflow management systems traditionally have humans as participants in the workflow. We showed that workflows are well suited for monitoring software agents.

(9) Rule-based negotiation. We showed that negotiation strategies and coordination information can be coded as rules, and that rule engine technology can be used to capture and exploit these rules. The benefits of such an approach are evidently the modularity of the rules and the possibility to edit them before and during the negotiation process.

(10) General contributions to e-negotiations research and to the visibility of the TEM project:

a. Active participation in three e-negotiations workshops[15], and in an e-negotiations seminar[16],

b. Membership in the e-negotiations group[17]. As results, contributions to the group classification for e-negotiations (work in progress), and involvement in the group's publications in a special issue on e-negotiations of the Group Decision and Negotiation journal[18] (work in progress).

1.6 Thesis Overview

The remainder of this document is organized as follows. Chapter 2 is dedicated to the literature review and is divided into two parts. The first part covers negotiations and auctions, e-negotiations, agent-mediated negotiations, negotiation strategies, and negotiation systems. The second part tackles the underlying concepts and technologies of CONSENSUS, namely, negotiation support systems, workflow management, rule-based representation of coordination in Multi-Agent Systems, and rule-based representation of negotiation strategies. In Chapter 3, we evaluate the different mechanisms used to describe negotiation protocols, and we introduce and justify our own representation. Chapter 4 details Combined Negotiations by covering the issues surrounding them and the difficulties they generate, and then presents the architecture of CONSENSUS, our Combined Negotiation Support System, along

[15] DEXA e-negotiations workshops (2000, 2001, 2002). http://www.dexa.org
[16] eNegotiations and eMarkets Seminar: http://johnmolson.concordia.ca/gkersten/enego2.htm
[17] E-negotiations Group Web page: http://enegotiations.wu-wien.ac.at/

with the details of its implementation. Chapter 5 details the design of our own negotiation platform GNP, and shows how it was reengineered from its ancestor GEE. In Chapter 6, our rule-driven approach for defining the behavior of negotiating software agents is presented, and results of the validation work are given. Finally, in Chapter 7, an assessment of our research and the major results we accomplished are presented, along with a discussion of future points of interest.

[18] Group Decision and Negotiation Journal. http://www.kluweronline.com/issn/0926-2644

CHAPTER 2

REVIEW OF THE LITERATURE

Preface

In this chapter, we review the state of the e-negotiation literature, with a focus on the issues that are most related to our research. The chapter is organized as follows:

The first part is dedicated to e-negotiations, as we set the stage to understanding combined negotiations and the issues surrounding them. The classification of e-negotiations is one of the stated goals of the E-negotiations group [ENe02] in which we actively participate. Our interest in classification is dictated by the need to understand the negotiation process in general, to isolate similarities between different negotiation protocols, and, among other things, by our involvement in building TEM's generic negotiation platform. The classification by the E-negotiations group will be presented in its current state, after we briefly introduce other classifications. Agent-mediated negotiations and negotiation strategies will then be covered, as they constitute a promising research area aiming to achieve automation by having software agents negotiate on behalf of a human consumer, basing their behavior on user-supplied strategies. Finally, our main concern, providing software support to the negotiation process, happens to be a focus of interest of both the academic and the business communities. Therefore, negotiation systems will be introduced, and a distinction will be made between negotiation servers and negotiation applications, in order to prepare the reader for Chapter 5, which presents our own negotiation server.

The second part introduces the concepts and technologies on which we rely in the design of our solution. The complexity of Combined Negotiations calls for tools to support the user in conducting them. Negotiation Support Systems (NSSs) have been

investigated long before the emergence of e-negotiations, so, naturally, we strive to leverage them. Our solution is built around a workflow management system used to model and enact the Combined Negotiation, thus an introduction to this important concept is necessary. Software agents will be in charge of negotiating on behalf of the consumer, therefore, their behavior needs to be represented accordingly. A rule-based approach is used to represent, manage, and study the agents' behavior, which we divide into coordination and strategies. A discussion of this approach will therefore be provided.

The reader will notice some redundancy in covering the literature throughout the thesis. This is due to the fact that the publications were made separately, at different phases of the project, and each one had to include a literature review of the concepts it presented.

2.1 E-Negotiations Related Issues

2.1.1 Definitions

Negotiation

Negotiation takes place when, based on offers made in the information phase, an agreement cannot be reached; or the agreement has potential for optimization, and the parties intending to carry out the transaction want to discuss their offers [Str99]. The information phase in this case is the one that precedes the negotiation, and during which participants gather information about products, other participants, etc. This phase is the equivalent of the first two steps of the CBB model introduced in Chapter 1. A negotiation mechanism is essentially a protocol within which agents interact to determine a contract, and most agree that auctions constitute a general class of such protocols [WWW98a].

Auction

McAfee and McMillan [MM87] define an auction as "a market institution with an explicit set of rules determining resource allocation and prices on the basis of bids from the market participants." For negotiations and auctions, the benefit of dynamically negotiating a price for a product instead of fixing it is that it relieves the merchant from needing to determine the value of the good a priori. Rather, this burden is pushed to the marketplace [MGM99].

Multi-dimensional Auction

There is more to auctions than just resolving the price. According to Reeves et al. [RGW+99b], auctions are mechanisms for determining price and other terms of an exchange. It is possible to define multi-dimensional generalizations and variants that resolve multiple issues at once. This can range from the simple approach of running independent one-dimensional auctions for all the parameters of interest, to more complicated approaches that directly manage higher-order interactions among the parameters. Even more complex in our opinion are combinatorial auctions, which

allow bidders to express offers for combinations of goods, and to determine an allocation maximizing overall revenue [RGW+99b].

Fixed price, Negotiation and Auction

According to Mester [Mes88], there are three basic mechanisms for price determination.

(1) A *fixed price mechanism* that is easy and inexpensive to manage, but lacks flexibility and does not reflect subtle variations in quality among different units of the same item.

(2) A *negotiated price mechanism* (i.e., bargaining or haggling) that is rather expensive to manage and time consuming, where buyers and sellers influence the price, and all aspects of the product and situation are taken into account.

(3) An *auction mechanism* that falls in between the first two. The seller sets the rules, and the prices are determined by competition between potential buyers. Auctions are more flexible than fixed price because they reflect current demand conditions, the latest information, and the taste of the bidder. Auctions are less time-consuming than negotiations because the seller compares offers simultaneously, and remains passive while buyers determine the price. The rules of the auction can be seen as a commitment on the part of the seller, and they restrict the kind of offers buyers can make. Rules show how the price will be determined so that demand equals supply (i.e., the market reaches equilibrium).

E-Marketplaces

Electronic marketplaces (e-marketplaces, also called exchanges or e-hubs) aggregate buyers and sellers, creating marketplace liquidity and reducing transaction costs. E-marketplaces can focus on specific industries or markets (vertical hubs) or on specific functions or business processes (functional hubs) [KT01].

Online Direct Sales

In this model, also called electronic retailing (e-tailing), companies sell their own products through their own Web sites. Such sites are managed by the supplier, are limited primarily to the supplier's products, and differentiate the supplier's products from others. Multiple competing sites may sell similar products. Companies following the direct sales model execute online various functions such as order configuration, pricing, order placement, order tracking, and customer service [KT01].

E-Negotiation

E-negotiation (i.e., negotiation in electronic commerce) can be defined as the process by which two or more parties multilaterally bargain resources for mutual intended gain, using the tools and techniques of electronic commerce [BS97].

E-Auctions (Online Auctions)

The use of auctions for exchanging goods such as artwork, antiques, agricultural produce, mineral rights, etc. has a long history [KT01]. With the advance of the Internet, the use of auctions for exchanging goods between individuals and companies has increased significantly. In addition to C2C online auctions such as eBay.com, there are two types of B2B online auctions: (1) *third-party auctions*, which allow companies to sell in e-marketplaces; and (2) *private auctions*, which companies build on their own extranets to serve their dealers or customers.

The most used online auction formats include the English auction (with variations on minimum bids, reserve prices, and buyout prices), first and second-price sealed-bid auctions, Dutch auctions, and continuous double auctions (CDAs).

Online auctions have several benefits compared to traditional auctions, including lower information, transaction, and participation costs; increased convenience; asynchronous bidding; and access to larger markets [Luc99].

Business-to-Business Negotiations

According to Lo et al. [LK99], transactions conducted on the Internet include retail with electronic shopping baskets and auctions, but negotiations that are typical of business-to-business (B2B) commerce have not yet gained enough attention. There is actually an interesting debate on whether auctions can replace negotiations on the Internet. In their paper titled "Are all E-commerce Negotiations Auctions?" Kersten et al. [KT00] answer their question with "no" and conclude that: "auction-like protocols will play a major role in contexts in which the determination of value is the primary concern. However, in B2B commerce, the participants are often less concerned with price and more with relationships. Negotiation-like protocols will dominate in these circumstances". Commenting on the "hype" surrounding auctions on the internet, the authors argue that [KT00]: "once the excitement is over, the reality that strikes is that business relationships and economic models are no simpler today than they were before the Internet, and different business models will need many different kinds of negotiation protocols, some of which will be very complex and rich in human factors."

2.1.2 Classifications

2.1.2.1 Justification

In practice, and due to a lack of comprehensive guidance, the design of negotiation systems has been evolving as a trial-and-error game [NBB+02]. It is therefore not surprising that many scientists seek to provide practical methods and processes to enrich the design process. One step towards that goal is the development of a negotiation classification. In this section, we justify the need for such classification, review some existing classifications, and present our own "London Classification" proposal.

There are two ways that the design process can be supported by the elaboration of a negotiation classification [NBB+02]. First, the analysis of negotiation scenarios can be improved. Different negotiation scenarios that belong to the same group share the

same characteristics, and are therefore expected to work in the same domain. Second, the selection of the most adequate negotiation scenario out of the space of possible negotiations (usually referred to as the negotiation space [LMJ00]) can be significantly streamlined. Instead of investigating a rather large number of possible negotiation scenarios, it is sufficient to observe a single class scenario.

Furthermore, based on a classification, standardized descriptions of the negotiation process can be exchanged over the network for consultation by human and software agents, before they engage in the negotiation [BK00a]. In such a case, software agents may be automatically instantiated using the negotiation description. This may eventually allow for a meta-negotiation, i.e., the participants can dynamically agree on a specific negotiation scenario.

2.1.2.2 Review of Existing Classifications

The 4D model presented in Subsection 1.2 constitutes a classification that deals with only one aspect of e-negotiations, namely, the item being negotiated. There are obviously more aspects to e-negotiations as we will see shortly.

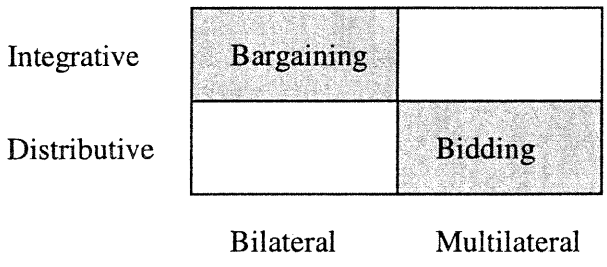

Figure 2.1: Simple Classification [Str99]

One of the many classifications we came across in our readings classifies negotiations as either *distributive* or *integrative*, and either *bilateral* or *multilateral* [Str99], [Str99b]. It defines bargaining as bilateral integrative negotiation, and bidding as multilateral distributive negotiation [KT00] (see Figure 2.1).

Another classification, given by Oliver [Oli97], is a four dimensional framework for e-commerce (clearly, the framework applies more to e-negotiations than to e-commerce). The dimensions are:

- *Negotiation versus posted-price*: this dimension indicates whether there is a possibility to negotiate or not. At one extreme, everything is negotiable (i.e., the bazaar approach), and at the other extreme, the prices are fixed (i.e., the department store approach).
- *System autonomy*: the system supporting the negotiation can have two extreme levels of autonomy. It can go from a fully autonomous mode, where there is no human intervention at all, to a helper/advisor mode where the user initiates actions and takes decisions.
- *Number of parties*: the minimum number of negotiating parties is two (bilateral negotiation). The number can be greater than two, leading to multilateral negotiation.
- *Number of issues*: the number of issues (i.e., negotiable attributes of the item) ranges from one issue (e.g., the price) to many issues.

Wurman et al. [WWW98a] give yet another classification. They describe a negotiation mechanism as "a protocol with which agents interact to determine a contract," and define auctions as "a general class of such protocols." They categorize an auction as:

- *Single or double*: in a single auction the bidders are either of type "buyers" or of type "sellers". In a double auction we can have multiple "buyers" and multiple "sellers" bidding in the same auction.
- *Sealed-bid or out-cry*: in the first category the bids submitted by the participants are not known until the auction closes. In the second one the bids are made public at the time they are made.
- *Ascending or descending*: in the ascending auction the bids begin low and keep increasing until a deal is made. In the descending auction, the seller (or the

auctioneer) begins with a higher price and lowers it continuously until someone bids on it.

Lomuscio et al. [LMJ00] propose a classification they call a "negotiation space". The dimensions of such a space are:

- *Cardinality of the negotiation*: this dimension is further divided into negotiation domain (single-issue or multiple-issue), and interactions (one-to-one, many-to-one, and many-to-many).
- *Agent characteristics*: agents are the (human or computational) entities that participate in the negotiation process. Agents are characterized by their role, rationality, knowledge, commitment, social behavior, and bidding strategy.
- *Environment and goods characteristics*: the negotiation environment can be static (i.e., variables such as prices do not change over time) and dynamic (variables do change over time). The characteristics of the goods can be private if the good is intended for private use (thus, its value is a personal matter), or public if its value depends on how the other agents value it.
- *Event Parameters*: this dimension deals with the way the offers and other events of the negotiation process are regulated. The model distinguishes the following: bid validity, bid visibility, clearing schedule and timeouts, and quotes schedule.
- *Information parameters*: to distinguish between price quotes and transaction history.
- *Allocation Parameters*: used to determine the winner of a auction if many agents have shown an interest in the good.

For more details on this classification, see [LWJ00].

2.1.2.3 The London Classification

In our meeting at the DEXA conference in Greenwich, London, in September 2000, the e-negotiations group [ENe02] decided to come up with a more complete classification framework for e-negotiations, and we took part in that effort. The

classification had to satisfy the following three goals: (1) provide a common set of terms describing e-negotiations with a well-defined set of classification criteria; (2) help analyze and understand the dimensions of e-negotiations and their interdependencies; and (3) assist the conceptual design of specific e-negotiations and support the abstraction necessary for the development of generic e-negotiation engines.

The resulting framework is commonly referred to as "The London Classification." Four aspects of e-negotiations were identified: the "people" aspect which deals with the participants in the negotiation, the "goods" aspect which is dedicated to the object of the negotiation, the "process" aspect which is concerned with the negotiation protocol to be followed by the participants, and finally the "evaluation criteria" aspect which covers the ways to evaluate a negotiation process. Our major contribution to this framework is the "process" aspect. Following is a description of that framework.

(I) The "People" Aspect

- *Number of parties:* There can be two or more parties participating in a negotiation. We talk of bilateral or multilateral negotiation, respectively.
- *Bidding activity*: A negotiation can be single-sided meaning that only buyers or only sellers are allowed to submit bids, or it can be double-sided meaning that buyers and sellers are both allowed to submit bids.
- *Admission*: A negotiation can be seller-open (buyer-open) meaning that there is no restriction on the admission of sellers (buyers) to participate in the negotiation. It can be seller-closed (buyer-closed) meaning that there exist restrictions on the admission of sellers (buyers) to participate in the negotiation.
- Collusion: A negotiation can be collusive or non-collusive. Collusion refers to agreements between buyers and/or sellers in order to achieve mutual benefits.
- Anonymity: A negotiation can be anonymous or non-anonymous. It is said to be anonymous if and only if any assignment of participants to their offers and the

identification of participants is impossible based on the information exchange provided by the negotiation protocol, i.e., on the endogenous information.

(II) The "Goods" Aspect

- *Number of items*: Items can be goods or services. We talk of a single-item negotiation if only one item is subject to the negotiation. We talk of multi-item negotiation if more than one item is subject to the negotiation.
- *Number of attributes*: One can negotiate on one attribute (e.g., the price), and thus we talk of single-attribute negotiation, or on multiple attributes of a deal (e.g., price, quality, terms of delivery), and thus we talk of multi-attribute negotiation.
- *Homogeneity*: In the case of multi-item negotiations, all items can be homogeneous (identical) or heterogeneous (non identical).

(III) The "Process" Aspect

- *Reverse versus forward*: We talk of a reverse auction when a buyer posts a request for an item, and sellers compete for the best conditions at which the buyer will accept the item (the bids go down). In a forward auction the seller lists an item for sale, and buyers post their bids (the bids go up). At any point the seller can accept one (or more) of the existing bids in which case the negotiation ends.
- *Single-phased versus multi-phased*: In a single-phased negotiation the rules are the same from the beginning to the end of the negotiation. If we allow for rules to change, then, each time they do, a new phase in the negotiation is started. We therefore talk of a multi-phased negotiation. As an example, a negotiation can start as a descending auction, then, whenever one participant bids to take the item, the rules change (a new phase begins), and the auction becomes an ascending auction.
- *Single-stage versus multi-stage*: If all the attributes of the item are negotiated at the same time, then we talk of a single-stage negotiation. In a multi-stage negotiation the parameters of the negotiation (for instance the user preferences or

the information available) can be changed after a stage, thus enabling another round of bidding.

- *Synchronized versus sequential versus combinatorial*: In a synchronized (also called simultaneous or parallel) auction there are n items. A participant can make m distinct bids on m distinct items (m not greater than n). The m bids are made simultaneously. The auction usually runs multiple rounds of sealed bids, announcing the bids after each round. The Federal Communication Commission (FCC) uses such a format to auction licenses for Personal Communication Services (PCS) [MM86]. If the n items are auctioned individually one after the other, then we talk of a sequential auction. If the auction allows for a participant to make a single bid for m items (m not greater than n), then we talk of a combinatorial (also called bundled) auction.
- *Open cry versus sealed bid*: In an open cry auction the bid (bids) made by a participant is (are) known by the other participants. Depending on the type of the auction, the bids can either go up or down. In a sealed bid auction, the participants make one secret bid (in some cases many secret bids). At the close of the auction, the winning bid is determined and announced (possibly with the identity of the winner). In a sealed bid auction, the bids can go up and down, and we say that the movement of the bids is haphazard.
- *Information revelation*: This deals with the transparency of the market and the amount of information available in the negotiation process (see open cry versus sealed bid). This covers also information such as, for instance, the identity of the other negotiating parties and the ranking or utility of a bid from the bid-takers perspective (in the multi-attribute case).
- *Agent-mediated versus manual bidding*: Software agents can be given a negotiation strategy and then be put in charge of a negotiation on behalf of the human participant. We talk of agent-mediated negotiation. Depending on their complexity, they can be totally autonomous, or they can just carry out simple tasks like bidding until a certain amount is reached with a certain increment. In

manual bidding, the participant makes all the decisions and submits the bids manually. In both cases a Negotiation Support System (NSS) can be used to enhance the chances of a good deal for the participant.

- *Discriminative versus non-discriminative*: Usually, once the bidding phase is over, the bidder with the highest bid gets the item being auctioned, but the price she pays could be equal to her bid or lower. In a Discriminative Auction (also known as Yankee Auction), the winners pay what they bid. In a non-discriminative auction, people with winning bids pay the amount of the lowest winning bid (for the sale of multiple similar items). Finally, in a Vickrey Auction (also referred to as second price sealed bid auction) the winner pays the price bid of the second highest bidder (for the sale of a single item).
- *Non-Repudiation*: In a negotiation, the participants are allowed or not to break their commitments. This means that bidders can or cannot repudiate (withdraw) submitted bids.

(IV) The "Evaluation Criteria" Aspect

- *Incentive compatibility*: Each negotiating agent is motivated (it is a dominating strategy for the agent) to reveal true preferences and to bid truthfully irrespective of the behavior of other agents. No advantages can be gained by modeling other agents or having additional information.
- *Computational complexity*: This refers to the complexity of the algorithm used to determine the winner(s) and/or the solution to a negotiation problem (price and feature discovery, allocation of quantities etc.). The complexity can vary from simple schemes such as "highest-bid-wins-at-second-price" to the NP-complete resolution problem of combinatorial auctions.
- *Convergence*: When designing a negotiation mechanism, it is important to know, if it converges towards equilibrium and if it produces a uniquely determined allocation. Negotiation mechanisms can explicitly be designed to converge.

- *Speed of convergence*: This is the time or number of stages needed until the negotiation mechanism converges towards equilibrium, meaning that no agent, given the current information, desires to change the resulting solution.
- *Stability*: A solution of a negotiation mechanism is stable if there is no subset of agents that could have done better by coming to an agreement outside the mechanism.
- *Integrative versus distributive*: A negotiation process is distributive (of win-lose nature) if a gain for one agent is necessarily a loss for the other agent (negotiations with single as well as multiple attributes can be distributive). In integrative (win-win) negotiations, joint gains are possible either through simultaneous improvements (trade-offs) based on opposing preferences or through the addition/invention of new attributes/options during the negotiation process.
- *Efficiency*: This property refers to the solution of the negotiation problem. A solution is efficient if, depending on the preferences and bids of the negotiating agents, there is no other allocation that is better for some agent without being worse for another agent.
- *Fairness*: A solution to the negotiation problem or the process involved to identify that solution is fair if it is not more beneficial (advantageous) to either selling or buying agents. Hence no agent envies any other agent. In a more narrow definition, a fair solution can also comprise that all agents involved think they received the same fraction of the total value (piece of the cake to divide) - which is, of course, difficult to assess. Fairness combined with efficiency ensures that the total value is distributed to everyone's satisfaction.
- *Correctness*: The following properties must be satisfied to achieve correctness: Only eligible bidders can submit bids, no one can impersonate a bidder, valid bids cannot be altered/eliminated by the auctioneer, bids are valid only for the specified auction, and the winner of the auction is always the highest/best bidder.

2.1.3 Agent Mediated E-Negotiations

According to Kephart et al., over the course of the next decade, the global economy and the Internet will merge into an information economy bustling with billions of autonomous software agents that exchange information goods and services with humans and other agents [KHG00]. Software agents can be defined as entities that execute functionalities in an autonomous, proactive, social, and adaptive fashion. These functionalities include searching, comparing, negotiating, and collaborating [Jon99], which make them particularly useful for the information-rich and process-rich environment of e-commerce [MGM98]. Thus, we talk of agent-mediated e-commerce. Software agents are also capable of personalized evaluation, complex coordination, and time-based interaction. They can be mobile, i.e., move between different computers or reside on one computer, and may either have learning capabilities or base their actions on pre-defined rules of behavior [LK99].

Agent-mediated e-commerce enables cheap negotiation between buyers and sellers on the details of an individual transaction – product features, value-added services, financing, and price. More specifically, agent-mediated negotiation may absorb many of the costs and inconveniences of negotiation [PUF99], and it is becoming more and more evident that agent-mediated negotiation stands on its own as an important field of study, quite apart from its eventual importance in the information economy [KHG00]. According to Rodriguez et al. [RMN+98], agent-mediated auctions appear as a convenient mechanism for automated trading, due mainly to the simplicity of their conventions for interaction, but also to the fact that online auctions may successfully reduce storage, delivery or clearinghouse costs in many markets.

E-negotiation, sometimes referred to as *automated negotiation*, takes place when the negotiating function is performed by (networked) computers. We talk of *fully automated* e-negotiation when all parties involved are software agents, *semi-automated* e-negotiation when a human negotiates with a software agent, and *manual* e-negotiation when all parties are human [BSS96].

Parkes et al. distinguish between *autonomous* and *semi-autonomous* agents. An autonomous agent requires a complete set of preferences in order to represent the user correctly in all situations. On the other hand, a semi-autonomous agent will bid on behalf of the user when it has enough knowledge to proceed, and query the user when its best action is ill-defined given the current information [PUF99]. A further distinction is made between a *reservation-price* agent and a *progressive-price* agent. The first type is an autonomous agent that places bids up to the value of a fixed reservation price. This is appropriate for users who have precise valuations of the auctioned goods. The second type is initialized with a lower bound and an optional upper bound on the true reservation prices of the user [PUF99]. This type is appropriate when a precise valuation of the good is not available. Note that a reservation-price agent is also referred to as a *proxy-bidder* by some commercial auction sites such as eBay [Eba02]. Finally, software agents can be involved in *competitive negotiations*, meaning that they are adversaries with conflicting interests. They can, on the other hand, be involved in *cooperative negotiations*, and in this case, they all aim at satisfying the same interest [BS97].

Software agents may facilitate all facets of electronic commerce, including shopping, advertising, negotiation, delivery, and marketing and sales analysis [KHG00]. *Pricebots*, for instance, are software agents that employ price-setting algorithms in an attempt to maximize profits, thus helping sellers to increase flexibility in their pricing strategies. An example taken from [GK99] points to *books.com* which uses a pricebot to monitor prices on competitor sites and offer the customer a lower price than what the competitors ask for. This is called real-time *dynamic pricing,* and it is clear that it can benefit enormously from automation. *Shopbots*, to cite another example, are software agents that automatically gather information from multiple online vendors about the price and quality of consumer goods and services [GK99]. Some successful shopbots on the Web today are *mysimon.com* and *bargainfinder.com.*

In our research, we are interested in the negotiation phase of an e-commerce transaction, and we intend to investigate, and eventually support many of the agent

features described above. We are aware of the challenges of automated negotiations and agree with Beam et al. that negotiation is difficult, and automated negotiation is even more so [BS97].

2.1.4 Negotiation Strategies

In this section, we first define negotiation strategies and examine the issues surrounding them. We then review some strategy-enabled infrastructures found in the e-negotiation research literature.

2.1.4.1 Definition and Issues

Negotiation is a form of interaction defined in terms of *protocols* and *strategies*. The protocols of the negotiation comprise the rules (i.e., the valid actions) of the game. For a given protocol, a bidder uses a rational strategy (i.e., a plan of action) to maximize her utility [GMM98]. Jennings et al. take this definition even further by identifying the following topics in automated negotiation [JFL+01]:

- *Negotiation protocols*: they cover the permissible types of participants, the negotiation states, the events that cause negotiation states to change, and the valid actions of the participants in particular states. Negotiation protocols dictate the types of operations that can be performed and influence the agents' strategies.
- *Negotiation objects*: they are the range of issues over which agreement must be reached. The object may contain a single issue (i.e., the price) or many issues (i.e., the price, quality, terms and conditions, etc.). Also in this topic are the operations that can be performed on the issues (i.e., accept or reject offers and counter offers, dynamically alter the object by adding or removing issues, etc.).
- *Decision making models*: they are used by the agents to act in line with the negotiation protocol in order to achieve their objectives. The relative success of two agents is determined by the effectiveness of their model – the better the model, the greater the agent's reward.

According to Wong et al. [WZK00], most current e-commerce systems use predefined and non-adaptive negotiation strategies (i.e., decision making models) in

the generation of offers and counter offers during the course of the negotiation. Commercial auction sites such as eBay, for instance, still require that consumers manage their own negotiation strategies over an extended period of time [MGM99]. There is however a possibility to relieve the user from managing the negotiation. EBay, for instance, offers the possibility of *proxy-bidding* which is actually a simple agent that uses a straightforward strategy: *bid until you reach a certain amount (the reserve-price), by going up each time with a certain increment (called the bid-increment)*. We should mention that this bidding technique has a disadvantage. To use it, one must reveal to the auction site the highest price one is willing to pay, giving the site information that could be used to cheat the bidder [Pri00]. As another example, the buying (selling) agents in the well-known KASBAH system [CM96] can choose between three negotiation strategies: anxious, cool-headed and frugal, corresponding to linear, quadratic, and exponential functions, respectively, for increasing (or decreasing) their bid (or ask price) for an item over time. Predefined and non-adaptive strategies are evidently not sufficient in regard to the ambitions of e-negotiations research.

Software agents participating in negotiations face tough decisions: whether or not to accept an offer, whether or not to make a final offer, whether or not to bid, how much to bid, etc. According to Gimenez-Fuentes et al. [GGR98], those decisions should take advantage of whatever information may be available in the market: participating traders, available goods and their expected re-sale value (in case the buyer intends to re-sell the good), historical experience on prices and participant's behavior, etc. The decisions are even harder to make due to the fact that this information is constantly changing and highly uncertain – new goods become available, other buyers come and leave, prices keep on changing, no one really knows for sure what utility functions other agents have, etc. Furthermore, the decisions should take into consideration the fact that the agents' goals, beliefs, and intentions change over time [NLJ96].

Some other information that is useful to the bidder is the valuation of the item at stake. In an auction where the market price is a common valuation among bidders

(e.g., an auction for personal computers), a bidder who wins the auction is the one with the highest, yet possibly overrated valuation. This is called the *winner's curse* [IFS+00]. Consequently, it is an advantage to know the real valuation of an item (i.e., its market price) in order to avoid the winner's curse. To do that, one might predict the market price of an item by monitoring its prices in several online auction sites [IFS+00].

Choosing a successful strategy by a negotiating software agent is not deterministic and depends on many factors, in particular on the strategies of the other competing agents [GGR98]. The reputation of the opponents can also be helpful in designing a winning strategy [KHG00].

Special care must be taken in the design of buying or selling artificial agents. According to Guimenez-Fuentes et al. [GGR98], designing, building, and tuning trading agents before letting them loose in widely competitive scenarios like electronic auctions inhabited by (both human and software) expert traders happens to be an arduous task. A buyer agent usually knows its owner preferences (i.e., her willingness-to-pay for an item) [Var95]. If a seller (human or artificial) of the item learns of the buyer's willingness-to-pay, it can make her a take-it-or-leave-it offer that will extract her entire surplus. This is not to the advantage of the buyer. On the other hand, artificial agents must guard against dynamic strategies that can extract private information. As an example, taken from [Var95], a seller agent knows its owner's reservation price and keeps it from the buyers. The seller (owner of the agent) would accept any offer that is higher than the reservation price. Suppose that a buyer agent (human or artificial) starts bidding at zero, and offers a sequence of incremental bids until it gets the item for a price slightly more than the seller's reservation price. This is not to the advantage of the seller.

Researchers are exploring various AI based techniques to provide adaptive behavior in the negotiation process. These techniques include logic, case-based reasoning (CBR), constraint-directed search, etc. [NLJ96]. The use of CBR is justified by the fact that good negotiation skills in humans seem to come from experience [WZK00].

Wong et al. use CBR as an approach to exploit past negotiation experience/strategies as guides to suggest suitable strategies to the current negotiation situation [WZK00].

Bayesian learning is used to make the agents learn negotiation strategies [ZS98]. According to Kephart et al., agents will have to learn, adapt, and anticipate, and in order to do so, they will use a variety of machine learning and optimization techniques [KHG00].

In our research, we take a rather pragmatic approach to the issue of negotiation strategies in automated negotiations. We see strategies as declarative (rule-based) knowledge that is given to the agents before or during the negotiation process. The benefits of this approach will be developed throughout this document.

2.1.4.2 Related Projects

Arguably, the most cited agent-mediated negotiation infrastructure is KASBAH [GMM98, CM96], an on-line multi-agent classified ad system centered on the negotiation phase of an e-commerce transaction. The user (seller or buyer) of this system creates an agent, gives it instructions and sends it to a centralized agent marketplace. The user has high-level control over the behavior of the agent. At the time of its creation, the user (a seller in this case) sets the desired price, the desired time to sell, the lowest acceptable price and the negotiation strategy. The negotiation strategy for sellers is based on the following rule: *Offer the item at the desired price. If there is no buyer, lower the asking price to generate more interest. At the desired date to sell, the asking price should be equal to the lowest acceptable price.* Three predefined strategies: anxious, cool-headed and frugal, corresponding to linear, quadratic, and exponential functions, respectively, can be used to set the rate for lowering (increasing in the case of buyers) the price.

Another well-known infrastructure, the AuctionBot [WWW98a], is a configurable auction server that allows human agents to create auctions. The server supports many auction types including the English, Dutch, and Vickrey auctions. Human agents can submit bids via Web forms, whereas software agents can submit bids using a special

API. Hu et al. designed an agent server [HRW99] that works on the user's behalf by submitting bids to the AuctionBot. The users specify the names of the auctions in which they want to participate, the initial amounts of the goods, and the bidding strategies they prefer. The agents then start bidding on the AuctionBot on behalf of the users. The agents keep bidding until the auction closes, and then report the results back to the users. The agent server runs three types of agents. *Competitive* agents that always bid their true reservation prices. The other two are strategic learning agents: a *price-modeling* agent that bases its next bid on the history data of clearing prices, and a *bidder-modeling* agent that models the actions of other agents by looking at the history data of those actions. The strategies are hardcoded into the agents.

The FM97.6 system, part of the Fishmarket project [RMN+98], is another pioneering test-bed for electronic auctions meant to provide support to software agent developers in defining, activating, and evaluating experimental scenarios (i.e., tournaments). These tournaments define standardized conditions under which software agents compete for maximizing their benefits [GGR98]. The tournaments are based on a downward bidding protocol (a variation of a Dutch auction) found in Spanish fish markets. Trading (buyer and seller) heterogeneous (human and software) agents of arbitrary complexity participate in these tournaments and are evaluated according to their market performance. Such competitive situations constitute convenient problem domains in which to study issues related to agent architectures in general and to agent-based trading strategies in particular [RMN+98]. The agents are provided with negotiating strategies based on two different approaches. A pragmatic approach provides buyer agents with heuristic guidelines. A more formal approach relies on possibilistic-based decision theory for handling possibilistic uncertainty on the consequences of actions due to the lack of knowledge about the competitors' behavior [GGR98].

A system called Experience Based Negotiator is presented in [WZK00]. It is made up of four functional components: (1) the Case-Based Negotiator assists users in negotiating with opponent agents by matching the current negotiating scenario with

previous successful negotiation cases and providing appropriate counter offers based on the best matched negotiation case; (2) the Case Browser allows users to browse a previous negotiation case repository; (3) the Statistics component supplies several useful descriptive statistics; and (4) the Case Maintenance component allows negotiation experts to moderate, maintain and update the case repository.

The approach taken by Su et al. [SHH00] is based on the idea that a consumer registers on a proxy negotiation server by giving a description of the goods or services she wants, her preferences, and a negotiation strategy. Then, the server takes over and looks for a supplier that matches the consumer. When it finds one, it starts a bargaining type negotiation until eventually a deal is reached. The server uses the negotiation strategy supplied by the consumer. A declarative approach is used to represent the negotiation strategies (see Section 2.2.4). Notice that in this case, the merchant brokering phase is the responsibility of the server, and that the user has no control over the negotiations once they are started. The user can only see the progress of the negotiations, but cannot intervene in them.

Other strategy-enabled test-beds do not focus on the negotiation phase of an e-commerce transaction. Researchers at IBM T. J. Watson, for instance, study pricebot dynamics by analyzing and simulating a series of price-setting strategies. For more on this research, see [KHG00].

2.1.5 Negotiation Systems

2.1.5.1 Introduction

The negotiation process is manually intensive; therefore, automating it, and capturing key information in a central repository that can be accessed by the negotiating parties, can reduce costs associated with the negotiation [Hur00]. Negotiation systems (also referred to as negotiation engines or platforms) are of great interest to both the business and academic communities. This interest is focused on the design process (among other aspects), with the aim of achieving higher efficiency and lower

transaction costs [Mal87], hopefully leading to more mature and successful electronic markets.

We dedicate this small section to introducing the reader to negotiation systems, leaving out Negotiation Support Systems[19] (NSSs), which we will cover in Section 2.2.1. GNP, the negotiation system developed in the TEM project, will be detailed in Chapter 5 of this thesis.

2.1.5.2 Negotiation Servers and Applications

There are two types of negotiation systems: negotiation servers and negotiation applications. Negotiation servers (i.e., engines or platforms) are typically systems that can run multiple negotiations, while negotiation applications usually provide only a single form of negotiation [Wri97]. Moreover, negotiation servers usually provide a workbench that can be used to generate an executable negotiation application. In this case, the special negotiation application constructed in order to fit within an existing market is regarded as a negotiation application.

The Michigan Internet AuctionBot[20] [WWW98a], implemented at the University of Michigan exemplifies the class of negotiation servers. This server is mainly designed to explore the possible auction design space. Consequently, it supports the generic configuration of various negotiation (auction) scenarios, according to the specifications of the negotiation initiator. The initiator can determine the concrete negotiation scenario consisting of the auction type, the bidding rules, the matching schedules, etc. For the other participants in the auction (i.e., the bidders), these settings are fixed. They are what we call the *negotiation protocol* throughout this document.

One system that is considered a "next generation" e-commerce server is the eAuctionHouse [EAu02] from Washington University. It is a configurable "auction

---

[19] Although we consider NSSs as negotiation systems as well, we chose to introduce them elsewhere to stress the fundamental differences between negotiation engines and NSSs, and consequently, between GNP and CONSENSUS.

[20] The AuctionBot, while still being cited in e-negotiation related publications, is no more supported by its creators.

house" that allows people across the Internet to buy and sell goods as well as to set up their own auctions. It does so by offering an expert system for setting up the auction by restricting the choice of auction types taking into account the auction settings (e.g., single or double auction, one or multiple items, one or multiple units of each item). It also supports combinatorial auctions with algorithms for winner determination.

Furthermore, eAuctionHouse makes it possible to bid via price-quantity graphs so bidders can express continuous preferences - when bidding, for instance, for a larger quantity, one might only accept a lower unit price. Another feature is the software agents that actively participate in an auction on behalf of the user while she is disconnected. More details can be found in [EAu02]

An example of a negotiation application is provided by the first and one of the most popular seller-driven online auctions, Egghead[21], formerly Onsale. Egghead operates a B2C auction site that auctions various consumer items such as vacation packages and sporting goods. The auctions use a single negotiation scenario, the Yankee format (the highest bidders win the merchandise at their bid price) with one or more identical items offered for sale at the same time. To place a bid one must: (1) enter the user name and password, the bid price per unit, the quantity; (2) select the shipping method; and (3) place the bid. Three bidding priorities apply when determining a winner: (1) bids are first ranked by price; (2) if bids are placed for the same price, bids of greater quantity take precedence over bids of lesser quantity; (3) if bids are placed for the same price and quantity, earlier bids take precedence over later bids.

The scheduled closing time for each auction is listed on the product page. However, if there is still bidding activity on a product during the final ten minutes, the auction will be extended into a special "Going...Going...Gone" period. The auction will close after ten consecutive minutes pass without further bid activity.

---

[21] http://www.egghead.com was recently taken over by http://www.amazon.com

Proxy bidding is available. The automatic bidding software enters the lowest possible winning bid for the user. If another buyer bids, the software will automatically raise the user's bid until she is winning or until she has reached her maximum bid.

Other popular negotiation applications are for example Ozro[22], eBay[23], and MOAI[24] LiveExchange.

2.1.5.3 Discussion[25]

Traditionally, negotiation applications have dominated negotiation design, but lately, the importance of negotiation servers has increased, and consequently, a need for configurable and generic negotiation engines is being felt. First, the dynamics of the market creates uncertainty concerning the market mechanism that fits the "real market". Thus, market design should allow for a relatively quick and easy configuration of negotiations reducing the overall setup costs. Second, the users' preferences are typically extremely heterogeneous, dividing the whole market into market segments. Moreover, there is no single negotiation design that fits all problems. A multiple negotiation design may provide all – or at least some – of the market segments with tailored trade mechanisms. An exhaustive guide to the choice of the appropriate mechanism does not and probably will never exist, making market design a challenging task. Third, from the software engineering perspective, negotiation servers are usually systems that consist of components with well-defined interfaces. Given the (short) limited lifecycle of these components, they should be easily exchangeable. Instead of rewriting the entire application, only a single (or a few) component must be replaced. If the core component – the negotiation engine – is domain independent, it can then be optimized in terms of scalability, security, and processing time.

---

[22] http://www.ozro.com
[23] http://www.ebay.com

2.2 CONSENSUS Related Issues

In this section, we review the concepts and technologies on which we base our solution. We discuss Negotiation Support Systems, workflow management issues relevant to CONSENSUS, coordination in MAS, and finally rule-based representation of negotiation strategies.

2.2.1 Negotiation Support Systems

Negotiation Support Systems research being rich and its literature abundant, we choose to address only the issues that are relevant to the design of our envisioned solution.

A Decision Support System (DSS) is a computer-based system used to assist and aid decision-makers in their decision making process [Ker99]. Several other DSS definitions can be found in [Ker99] along with some functional and performance requirements a DSS must have. We retained the following three requirements that we believe are applicable to our envisioned CNSS:

- A DSS is used to analyze a problem, determine alternative solutions, and select one;
- A DSS often requires user involvement in the construction of the problem representation;
- A DSS must be user friendly.

A Negotiation Support System (NSS) on the other hand is a software system consisting of two components [LK99]: (1) a Decision Support Component, and (2) a Communication Component. The first component enhances the information processing capabilities of the negotiators. It can, for instance, provide a quantitative evaluation of offers, show a graphical representation of the negotiation, maintain a history of the negotiation, etc. The second component facilitates the exchange of

---

[24] http://www.moai.com
[25] Taken from [NBB+02]

offers and arguments between the negotiating parties. The decision support and the communication components are both important to our vision of a CNSS.

According to Beam et al. [BSS96], NSSs are specially geared towards helping human negotiators making better decisions and negotiating more productively, and are a step towards automated negotiation. The authors go on saying: while NSS are quite powerful tools, they are far from able to support automated negotiations on their own. NSS require near-constant human input, and both the initial problem setup and all final decisions are left to the human negotiators. Our envisioned CNSS is not aimed at replacing the user, but it is rather meant to support her while keeping her involved in the negotiation process, regardless of the degree of automation it provides.

We agree that *support* should be the main functionality of a Negotiation Support System, however, we see *automation* as an important additional functionality that such systems should provide for the users.

The need for decision support for e-marketplaces can be extended to other phases of an e-commerce transaction. According to Keskinocak and Tayur [KT01], companies will need decision support in at least two areas: (1) identifying potential matches, and (2) deciding on which subset of the potential matches to execute. In our case, in addition to supporting the consumer in the negotiation process, the envisioned CNSS is meant to support her in identifying the items that respond better to her needs, and in identifying the providers of these items (i.e., the product and merchant brokering phases in the CBB model – see Chapter 1).

Through our envisioned solution, we bring NSS, software agents and Internet technologies to the negotiation process. The benefits, as reported by Kersten et al. [KL01], can be summarized as follows:

- NSS are designed to help and advise negotiators; they are used to structure and analyze the problem, elicit preferences and use them to construct a utility function, determine feasible and efficient alternatives, visualize different aspects of the problem and the process, and facilitate communication.

- Software agents automate mundane operations.
- Internet technologies reduce costs of both auctions and negotiations, and introduce new tools to access, conduct and analyze these processes.

2.2.2 Workflow Management[26]

2.2.2.1 Introduction

Complex tasks, oftentimes occurring in heterogeneous environments, require a certain structure (modeling) in order to facilitate their management as well as the automation of their execution. Because of its immense promise, workflow technology was introduced to deal with this kind of tasks. The Workflow Management Coalition (WfMC[27]) proposes the following definition of workflows, which is widely used within the literature: "a workflow is the automation of a business process - defined as a set of one or more linked activities, which collectively realize a business objective, in whole or in part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules" [WfM99]. Simply put, workflow technology aims to provide as much computer support as possible to the modeling, execution, supervision, and possibly reengineering of business processes [CHR+98, JB96]. Some examples of such processes 3 are: the processing of an insurance claim, the processing of a purchase order, the processing of customer support, etc.

A Workflow Management System (WfMS), on the other hand, is a piece of software that manages the workflow efficiently by tracking and controlling its execution. It provides support in three broad functional areas: (1) workflow definition (capturing the definition of the business process); (2) workflow execution (managing the execution of the workflow processes in an operational environment, sequencing the

---

[26] Based on [BBK01]
[27] http://www.wfmc.org

various activities to be performed) and; (3) workflow monitoring (monitoring the status of workflow processes).

2.2.2.2 Workflows and E-commerce

Workflow management is an active area of research, and its application to e-negotiations is novel. Muth et al., for instance, argue that workflow management is one of the enabling technologies in e-commerce, and that it should play a role in providing a highly dependable infrastructure for e-commerce [MWW98]. The authors see an e-commerce process as a three-phase scenario: the pre-sales phase, the sales phase, and the post-sales phase. In the first phase, the consumer performs a set of unstructured activities such as collecting information about the items she wants to purchase. In the second phase, the consumer requests sales offers, makes a decision, and places an order. In this phase, it is possible for the consumer to engage in a negotiation with the provider, using a well-defined protocol to be followed by both parties. Obviously, the execution of this protocol has to be highly dependable. The third phase consists of auxiliary activities either related to the purchased product or to the general relationship between the two parties. The whole three-phase e-commerce process can benefit from workflow technology. Capturing (i.e., modeling) the whole process or parts of it, for instance, will ensure more dependability.

2.2.2.3 Modeling CNs as Workflows

We aim to provide a CNSS to support the consumer in conducting the CN. The system should provide the means for modeling the CN by specifying the sequencing of the individual negotiations and the dependencies between them. Modeling the CN is a first step towards tackling its complexity, and towards ensuring that it will be conducted according to a well thought plan. We call this the modeling phase (or the build-time phase). The support tool will then be used to run the CN by enacting the model. We call this the enactment phase (or the run-time phase).

Reviewing the literature, we observe that special attention is dedicated to the techniques for scheduling, managing, and interfacing workflows to applications. Few

works, however, address the conceptual modeling of workflows [CKO92, BB00, LS97].

It is clear that, as workflow technology attempts to expand its applications, modeling issues are becoming more and more important. So what are the benefits of modeling in general, and modeling the CN in particular? A model is an abstract representation of the reality that excludes much of the world's infinite detail. According to Curtis et al., a model reduces the complexity of understanding or interacting with a phenomenon by eliminating the detail that does not influence its relevant behavior [CKO92]. However, many essential forms of information discussed in [CKO92], must be kept to adequately describe a model. Among these forms are "what is going to be done," "who is going to do it," "when and where will it be done," "how and why will it be done," and "who is depending on its being done." Modeling languages differ in the extent to which their constructs highlight the information that answers these questions. They usually present at least one perspective related to these questions. Curtis et al. suggest four common perspectives for modeling [CKO92]: (1) functional (the "what" question); (2) behavioral (the "when and how" questions); (3) organizational (the "where and by whom (which agents)" questions); and (4) informational (the structure of entities, and the relationships among them) modeling.

Five main objectives for models, which range from comprehensibility to enactability, were discussed in [CKO92]: (1) facilitation of human understanding and communication; (2) supporting process improvement; (3) supporting process management; (4) automating process guidance; and (5) automating execution.

To the extent that automation is involved, process representation becomes a vital issue in redesigning work and allocating responsibilities between humans and computers. This requirement reflects the growing use of distributed, networked systems to link the interacting agents responsible for executing a business process [CKO92].

A combined negotiation is indeed a complex process since it involves many agents, each one conducting an individual negotiation on a distant server while cooperating

with other agents in solving a common problem: "the consumer wants the whole package or nothing at the best price possible." Modeling a CN gives a visual representation, which is easily understandable by humans, and identifies and formalizes all the necessary items of the CN. This may be helpful in a prospective evolution or modification of the current negotiation items, their sequencing and the dependencies between them. Modeling the CN also incites to reason about its variables and attributes (such as the prices, dates, etc.). It may for instance specify some forecasting (such as "what will be the new reserve-price based on the outcome of the negotiations that are already done"). Monitoring, managing, and coordinating the CN are also objectives of the modeling process.

2.2.2.4 WfMS and Software Agents

Some applications are workflow-enabled and can be invoked indirectly by the workflow engine [HH99]. However, other applications are not compatible with the standardized workflow interface. Their integration into the business process is possible via an actor agent. The later takes the role of an actor, which is defined in the context of a WfMS, as being a resource that performs a task. It is invoked by a workflow engine, and enables indirect interaction of this engine and the application in question. At this prospect, the agents within CONSENSUS are first instantiated before being invoked by the workflow engine. An indirect interaction between the workflow engine and the negotiation servers is observed in that case.

2.2.3 Coordination in Multi-Agent Systems

2.2.3.1 Introduction

In this section, we examine the coordination problem in a Multi-Agent System (MAS). We briefly describe the existing coordination techniques and review the solutions implemented by some related projects.

When adopting an agent-oriented view of computation, it is readily apparent that most problems require or involve multiple agents [JFL+01]. The interaction between agents can vary from simple information interchanges, to requests for particular

actions to be performed, and on to cooperation (working together to achieve a common objective) and coordination (arranging for related activities to be performed in a coherent manner) [JFL+01]. Coordination is a central issue in software agent systems and, more generally, in distributed artificial intelligence (DAI). Multiple agents need to be coordinated for the following reasons [NGL96]: preventing anarchy and chaos; meeting global constraints; distributing expertise, resources or information; dealing with dependencies between agents; and ensuring efficiency [NGL96]. Nwana et al. identify the following four coordination techniques:

- *Organizational structuring*: an agent's responsibilities, capabilities, connectivity and control flow are defined by the organization. This technique can be implemented in two ways: (1) a *master/slave* approach where a master agent distributes work to the slave agents, (2) a *blackboard* approach where the agents post and read from a general blackboard. This approach is suitable if the tasks have already been assigned, a priori, to the agents. There may or may not be a need for a master agent in this case. Coordination can occur between peer agents.
- *Contracting*: agents can assume two roles. A manager who breaks a problem into sub-problems and searches for contractors to do them, as well as to monitor the problem's overall solution, and a contractor who does the sub-work.
- *Multi-agent planning*: agents build a multi-agent plan that details all the future actions and interactions required to achieve their goals. The solution can be centralized, with a coordinating agent that analyzes the other agents' plans and checks them for possible inconsistencies. The solution can also be distributed, and in this case each agent is provided with a model of the other agents' plan. Agents communicate in order to build and update their individual plans and their models of others' until all conflicts are removed.
- *Negotiation*: coordination can be seen as a negotiation problem. It can be defined as the communication process of a group of agents in order to reach a mutually accepted agreement on some matter.

2.2.3.1 Related Projects

One system that addresses the coordination problem is the Biddingbot of Carnegie Mellon University [IFS+00]. The Biddingbot is a multi-agent system that supports users in attending, monitoring, and bidding in multiple auctions. To predict a market price of an item, agents simultaneously monitor prices of the item in several auction sites. The Biddingbot MAS consists of one leader agent and several bidder agents. Each bidder agent is assigned to an auction site. Bidder agents cooperatively gather information, monitor, and bid in the multiple sites simultaneously. The leader agent facilitates cooperation among bidder agents as a matchmaker, sends user requests to the bidder agents, and presents bidding information to the user. A five-step cooperation protocol is used between the agents. For details see [IFS+00].

Another system, designed by Priest et al. at the HP Laboratory in Bristol, England, takes a totally different approach. Instead of using a MAS, they simply use a single agent [Pri00]. The agent aims to purchase one or more identical goods on behalf of its user. For that, it participates in many auctions for that good, and coordinates bids across them to hold the lowest bids. As auctions progress and it is outbid, it may bid in the same auction or choose to place a bid in a different auction. The algorithm used by the agent consists of two parts. First, it has a coordination component, which ensures it has the lowest leading bids possible to purchase the appropriate number of goods. Second, it has a belief-based learning and utility analysis component, to determine if it should deliberately *lose* an auction in the hope of doing better in another auction later.

Not directly related to e-negotiation research, but highly relevant in the way it deals with coordination in MAS is the ARCHON (Architecture for Cooperative Heterogeneous ON-line systems) project [BCL96]. ARCHON is a European DAI project started in the early 1990's. It focused on a general-purpose architecture, software framework, and methodology for supporting real world DAI. The agents in ARCHON are divided into two layers: an ARCHON Layer and an application program. The former encapsulates generic knowledge about cooperative interaction,

which is domain independent and encoded in terms of production rules. The latter is a special problem solving application, which performs the problem solving process. Reusable, generic cooperation know-how is encoded in an ARCHON layer by production rules of the form:

Rule-1

If an agent has generated a piece of information i
and it believes that i is of use to an acquaintance
then send i to that acquaintance

Rule-2

If an agent has a skill to perform and is not able to perform it
then locally seek assistance from another agent

ARCHON is important in making a separation between generic knowledge about cooperative interaction and application knowledge. Thus, its approach enables generic cooperation features to be reused for other applications.

In CONSENSUS, we adopt a similar approach for representing and managing the coordination of our software agents.

2.2.4 Rule-based Negotiation Strategies

In this section, we discuss two related approaches that inspired us in choosing a representation mechanism for negotiation strategies.

The first one is the work of Grosof et al. at the IBM laboratories [Gro97]. The research is concerned with the negotiation phase of contracting, and more specifically, with the representation of business rules in contracts. It relies on the fact that many contract terms involve conditional relationships, and can conveniently be expressed as rules, often called business rules. Some examples of such rules are [GLC99]:

Rule A	"If buyer returns the purchased good for any reason, within 30 days, then the purchase amount, minus a 10% restocking fee, will be refunded."
Rule B	"If buyer returns the purchased good because it is defective, within 1 year, then the full purchase amount will be refunded."
Priority Rule	"If both rules A and B apply, then Rule B wins."

The aim of the research is to design a shared language with which agents can reach a *common understanding* of rules in contracts, such that the rules are relatively *easily modifiable*, *communicable*, and *executable* by the agents [GLC99]. To that end,

several approaches for representing business rules are identified: (1) if-then code constructs within an imperative language such as C++ or Java, (2) declarative languages such as Prolog, (3) SQL views, (4) event-condition-action rules / "active rules" / "triggers", (5) production rules, and (6) Knowledge Interchange Format (KIF). A more detailed description and an evaluation of these approaches can be found in [GLC99]. None of the approaches cited above satisfies completely the requirements stated by Grosof et al., and so, a new approach called Courteous Logic was devised. It is an extension of ordinary logic achieved by adding the possibility to express prioritized conflict handling as rules. For a complete description of Courteous Logic, see [Gro97]. An example of a rule taken from [RGW+99a] gives an idea of the syntax used: "*A buyer can modify the departure time up until 14 days before scheduled departure if the buyer is a preferred customer*" is coded in Courteous Logic as follows:

```
<leadTimeRule1>
        modificationNotice(?Buyer, ?Seller, ?Flight, 14days) ←
        PreferedCustomerOf(?Buyer, ?Seller).
```

The "←" indicates "if" and the "?" prefix indicates a logical variable.

The second approach is the work of Su et al. at the University of Florida. Contrary to some existing approaches in which negotiation strategies are hardcoded in negotiation agent programs, they use a high-level rule specification language and GUI tools to allow human negotiation experts to dynamically add and change negotiation rules at run-time [SHH00]. Each strategy is expressed in terms of an Event-Trigger-Rule (ETR), which specifies the condition to be checked and the actions to be taken by the negotiation server. Rules are activated upon the occurrence of specific events during the negotiation process. Instead of a rule engine, they use an ETR server [LS98]. The server manages events and triggers rules that are relevant to the posted event. Consider the following example taken from [SHH00].

The rule: "If deliver_day is out of range, check the lower bound of the constraint for attribute deliver_day. If the lower bound is still greater than 10 (i.e., the bottom line),

shorten the delivery time by two days. Otherwise, the constraint is considered unResolvable and human intervention is required" is coded as an ETR as follows:

TriggerEvent

SupplierComputer_Systemdelivery_day

SR1:

Condition: getLowBound("delivery_day") > 10
Action: downLowBound("delivery_day", 2);
Alternative: unResolvable("delivery_day cannot be satisfied");

Rules have two main advantages over other software specification approaches and programming languages. First, they are at a relatively high level of abstraction, and are closer to human understandability, especially by business domain experts who are typically non-programmers. Second, rules are relatively easy to modify dynamically [RGW+99a]. It is widely recognized in modern software design that a vital aspect of modifiability is modularity and locality in revision. New behavior can be specified by simply adding rules, without needing to modify the previous ones. Furthermore, rules can be augmented with procedural attachments so that they have an effect beyond pure-belief inferencing. Procedural attachments are the association of procedure calls with belief expressions (e.g., the association of a Java method) [GLC99].

CHAPTER 3

DESCRIPTION OF NEGOTIATION PROTOCOLS

Preface

A basic finding of negotiation sciences is that there is not a single negotiation protocol for all possible negotiation situations; therefore different negotiation protocols are appropriate in different situations [Bic00]. The diversity of negotiation protocols (i.e., types, styles) evidently calls for a clear description of the rules that govern them. The participants in the negotiation process need to know the rules before engaging in a negotiation, and to this end, in an automated environment a mechanism is needed which allows for the rules to be serialized and visualized. Furthermore, this formal description should be executable, in order for the negotiation designer to simulate the negotiation process and to investigate its compliance with the underlying requirements.

In this chapter, we first discuss a list of requirements for a formalism that is appropriate for capturing negotiation processes. Second, we review five major techniques and formalisms for describing such processes. Third, we evaluate these formalisms according to our requirements and summarize our findings.

The evaluation led us to suggest UML statecharts as the best choice for representing negotiation protocols. They have a good formal basis and can be serialized, visualized, and executed. They are also well established, are easy to understand, are complete, and can be converted into other formalisms. Therefore, they enable the designer of negotiation protocols to understand, implement and test different negotiation types more thoroughly before making them available for general use (with or without software support).

The architecture of CONSENSUS (see Chapter 4) also relies on statechart descriptions of negotiation protocols to automatically instantiate the software agents that participate in the corresponding negotiations. Furthermore, the design of GNP (see Chapter 5) calls for leveraging the formal descriptions of negotiation rules to provide a repository of negotiation process descriptions. These negotiation processes could be described using statecharts, and the actual scripts that implement the negotiation (i.e., its economic rules) could be automatically generated from a statechart description, thus making GNP even more generic.

This study helped us understand the different negotiation types, and identify their common features. The UML statechart description enabled us to implement and test various auction (auctions are special cases of negotiations) protocols on GNP before using them as nodes in combined negotiations. In the context on CONSENSUS, we call "node" a negotiation where one single item of the package is negotiated.

The remainder of this chapter is a modified version of the following publication:

An Evaluation of Formalisms for Negotiations in E-Commerce, by Morad Benyoucef and Rudolf K. Keller. In *Proceedings of the Workshop on Distributed Communities on the Web*, pages 45-54, Quebec City, QC, Canada, June 2000. Springer. LNCS 1830.

An Evaluation of Formalisms for Negotiations in E-Commerce

Abstract. The diversity of negotiation types in e-commerce calls for a clear description of the rules that govern them. The participant has to know the rules before engaging in a negotiation, and to this end, a formalism is needed which allows for the serialization and visualization of the rules. Furthermore, this formal description should be executable, in order to simulate the negotiation process and investigate its compliance with the underlying requirements. As a consequence of such formalization, the negotiation process and the software supporting it may be separated. This paper discusses the requirements for a formalism that is appropriate for capturing negotiation processes. It then presents five major techniques and formalisms for describing such processes and evaluates them according to the requirements. The paper concludes that the Statechart formalism is most suitable for the negotiation types considered.

3.1 Introduction

According to the Object Management Group (OMG) and CommerceNet whitepaper on electronic commerce (e-commerce) [ECD97], commerce is at its heart an exchange of information. To obtain something of value from someone else you need to: (1) find a person who has what you want and communicate your desire, (2) negotiate the terms of a deal, and (3) carry out the deal. Each of these activities involves an exchange of information. Negotiation can be defined as "mechanisms that allow a recursive interaction between a principal and a respondent in the resolution of a good deal" [OMG99]. The principal and the respondent are usually a consumer and a supplier. In general, they need to negotiate the price, the delivery date, the conditions of the purchase, the terms of the guarantee, etc. If the negotiation is carried out automatically or semi-automatically, we talk of electronic negotiation (e-negotiation).

The diversity of negotiation types calls for a clear description of the rules that govern them. The participant has to know the rules before engaging in a negotiation, and to this end, a formalism is needed which allows for the serialization and visualization of the rules. Furthermore, this formal description should be executable, in order to simulate the negotiation process and investigate its compliance with the underlying requirements. Note that "negotiation rules" denote the rules governing the negotiation, whereas "negotiation strategy" describes the rules used by an individual participant during the negotiation in order to make the best out of her participation.

Our research focuses on *combined negotiations*, and we are currently studying concepts and architectures to support them [BK00c]. We will rely on formal descriptions of negotiation rules to describe, understand, and test the different negotiation types that make up a combined negotiation. We propose a tool called Combined Negotiation Support System (CNSS). Its architecture is based on workflow technology, negotiating software agents, and decision support systems. A workflow that models a combined negotiation will be constructed to reflect the constraints and dependencies that exist among the individual negotiations. Software agents will be instantiated and assigned to the individual negotiations. They will be actors in the workflow, and the user can monitor them via the administration tool of a workflow management system. The CNSS will be similar to a decision support system, by supporting the user in controlling and tracking the progress of the combined negotiation [BK00c].

This work is being conducted as part of the TEM (Towards Electronic Marketplaces) project. The project addresses market design issues in respect to resource allocation and control and reward mechanisms, investigates open protocols for electronic marketplaces, and explores concepts and tools for e-negotiations. As a common infrastructure of TEM, we are developing a generic negotiation platform (GNP) [BKL+00]. GNP is meant to support various types of negotiations. The GNP architecture assumes that the different negotiation types are described in a uniform

and formal way, and that the descriptions can be serialized for exchange in the e-marketplace.

The contributions of this paper are threefold. First, we discuss a list of requirements for a formalism that is appropriate for capturing negotiation processes. Second, we review five major techniques and formalisms for describing such processes. Third, we evaluate these formalisms according to our requirements and summarize our findings in tabular form.

Section 2 of the paper briefly introduces the concept of negotiation. Sections 3, 4, and 5 present the three main contributions of the paper. Finally, a short conclusion is given in Section 6.

3.2 Negotiations

"Negotiating takes place when, based on offers made in the information phase, an agreement cannot be reached or the agreement has potential for optimization and the parties intending to carry out the transaction want to discuss their offers" [Str99]. The information phase is the period that precedes the negotiation and during which the participant gathers information about products, market participants, etc.

A considerable research effort is being dedicated to the subject of negotiations [KF98a, KF98b, WWW98a, LG99, BS97, BSS96, MGM99, Tur97, WW98]. As described by Kumar and Feldman [KF98a, KF98b], the simplest form of negotiation is no negotiation at all (also called fixed-price sale) where the seller offers her goods or services through a catalogue at take-it-or-leave-it non-negotiable prices. Auctions are one simple form of negotiation and are at present the most visible type of e-negotiations on the Internet. Negotiations can take a more complex form called bargaining. It involves making proposals and counter-proposals until an agreement is reached or until the negotiation is aborted [SHH00]. The OMG considers bargaining as a bilateral or multi-lateral negotiation depending on whether there are two parties (one-to-one bargaining) or many parties (many-to-many bargaining) involved in the negotiation [OMG99]. Negotiations are further classified as distributive or integrative

[Str99]. In distributive negotiations, only one attribute is negotiable (usually the price). The parties have opposing interests, that is, one party tries to minimize (e.g., the buyer) and the other party tries to maximize (e.g., the seller) the price. In integrative negotiations, multiple attributes of the item are negotiable. Furthermore, combinatory negotiations [San99b] are a form of negotiation that involves making bids on combinations of goods or services, and one of the main challenges is for the auctioneer to determine the winning bid. In a combined negotiation, finally, the user engages in many negotiations for different goods or services. The different negotiations are independent from each other, whereas the goods and services are typically interdependent. A more detailed and complete description of negotiation types can be found in [Ben00].

3.3 Requirements for Negotiation Formalism

Before we discuss formalisms and techniques for describing the variety of negotiation types presented in the above section, we need to define the criteria we will use in evaluating these approaches. We have come up with a non-exhaustive list of eight interdependent criteria.

3.3.1 Formal Basis

If we are to make buying and selling decisions based on automated negotiations, then it is important that we have high confidence in the software involved in this activity. Cass et al. [CLL+99] suggest that an automated negotiation must have four necessary properties. It must be: (1) *Correct*: an automated negotiation must not contain errors such as deadlocks or incorrect handling of exceptions; (2) *Reasonable*: it must allow adequate time for bids to be made; (3) *Robust*: it should continue to function properly after an improper action by the user; and (4) *Fast*: it has to execute and respond quickly. We believe that there should be a fifth property. An automated negotiation must also be *traceable*. In order to be trusted, the software must be able to justify its actions to the user if necessary. This may be done by tracing the execution and by showing the decisions taken together with their rationale. In order to achieve these five properties, we need to formally describe negotiation processes. We agree with

Cass et al. [CLL+99] that "a notation with well-defined, well-formed semantics can facilitate verification of desirable properties". Formalization will enable us to separate the process of negotiation from the other parts of the software. We believe that the rules governing the negotiation should not be hardcoded. The software tool supporting the negotiation should be able to pick one type (style) of negotiation from a repository and use it. This will lead to efficient implementation and easy testing and, last but not least, will encourage reuse. We can actually benefit from the fact that negotiation processes contain parts that are common to all of them.

3.3.2 Serialization

Negotiation rules should be known to the participant (human, software agent, proxy negotiation server, etc.) and to anyone who wants to consult them before engaging in the negotiation. Their rendering should be intuitive and flexible in order to cope with various audiences. We believe that the rules are as important as, or perhaps even more, than the other information describing the good or service that is the object of the negotiation. We agree with Wurman et al. [WWW98a] that "implementing auction mechanisms in a larger context will require tools to aid agents in finding appropriate auctions, inform them about auction rules, and perform many other market facilitation functions". It should therefore be possible to serialize the negotiation rules and transfer them over the network.

3.3.3 Visualization

Visualization (graphical if possible) is important to the user of the negotiation tool because humans are known to understand visual information better than textual information. Furthermore, animation possibilities can make the execution (if supported) more attractive and easier to follow by the user. The developer of the negotiation tool should also be able to visualize and eventually animate the negotiation rules, for instance for prototyping and testing purposes.

3.3.4 Executability

If the formalism chosen to describe the negotiation rules is executable, the negotiation process can be simulated. Simulation may support the verification of the formal description in respect to correctness, consistency, completeness, absence of deadlocks, and alike. Furthermore, it will help analyzing and validating the description against the underlying requirements.

3.3.5 Other Criteria

A good description approach should be *well established* for at least two reasons: first, it will already have passed the test of time, and, second, software tools supporting the approach may be available. Moreover, it should be *easy to understand*. This criterion is important for the person who consults the negotiation rules before engaging in it. This consultation should be effortless and quick. If it is not, the user might dislike the negotiation tool, or even worse, she might use it without really understanding it. The description must also be *complete*. The formalism chosen to describe the negotiation rules should allow for capturing all the details of the negotiation type at hand. Otherwise, we would have to complement the description using natural language or another formalism. Finally, we believe that it should be possible to *automatically convert* the description to other existing formalisms. This criterion suggests the smooth transfer of the description to platforms supporting other formalisms.

3.4 Candidate Formalisms

We have identified five major description techniques and formalisms for negotiations. They are described below.

3.4.1 Natural Language

Bidding is a popular form of negotiation. Among the many definitions we found the following one [SHH00]: "bidding is when the consumer specifies what she wants and the suppliers make their bids. Based on their bids, the consumer selects a supplier from whom to order the good or service". This definition is correct, but it can be confusing if we compare it to an alternative definition of the same term. In fact, it is

widely known that "bidding" means "proposing a price on an item, and committing to take the item for that price if the seller agrees, or, in the case of an auction, if there are no higher bids than the proposed price". We can see that the use of natural language leaves the door open to dangerous ambiguities. It is this and other examples that call for a formal description of negotiation processes.

3.4.2 Agent Coordination Language

Little-JIL is an agent co-ordination language realized at the University of Massachusetts at Amherst. Programs written in Little-JIL describe the coordination and communication among agents, which enables them to execute the process in question [Wis98]. It is a graphical language in which processes are broken down into steps. The steps are connected to each other by edges that represent control flow and data flow. Each step is assigned to an agent to execute. A program in Little-JIL has a root that represents the entire process, and the root itself is decomposed as necessary into steps. Visually, a step is a rectangle with many tags associated to it (the name of the step, the resources necessary to execute it, the parameters, the exceptions that the step might raise, etc.).

Little-JIL was used to describe a set of negotiation processes [CLL+99]. The resulting programs define the coordination and communication between agents involved in the negotiation. An agent might be a seller or a buyer, and each agent is assigned a number of steps to perform. Describing a negotiation process in terms of steps and edges between them simplifies reasoning by following the flow of control and the flow of data in the tree of steps. Since a negotiation typically involves multiple participants, it is convenient to model it using Little-JIL by decomposing it into steps to be carried out by the participants and by providing coordination and communication between them.

A sealed-bid auction and an open-cry auction are described in [CLL+99] using this mechanism. The authors claim the following benefits from using Little-JIL: ease of understanding, the possibility to compare negotiation processes, to analyze them, to separate the process of negotiation from the application, and to execute a negotiation

process. The ease of understanding is not as evident as the authors claim. The notations are quite exotic and the bigger the process gets, the harder it is to understand what it does. We do not know if a description using this formalism can be serialized or converted to other formalisms, but we do know that it is complete.

3.4.3 Finite State Machines

Kumar and Feldman [KF98a] use Finite State Machines (FSMs) to model a negotiation. The states of the FSM are the states of the negotiation, and its input alphabet is the set of messages sent by the participants. A message is expressed as a pair < p , m > where p is the sender of the message and m is the message sent out. The output alphabet of the FSM is the set of messages sent to the participants. These messages are expressed as pairs << p , m >> where p is the subset of all the participants that will receive the message and m is the message itself. The process flow of the negotiation maps into the transitions of the FSM. The messages make the negotiation go from one state to another. Figure 3.1 gives a description of an English auction using this notation. The four states of Figure 3.1 can be found in the FSMs describing most of the other types of negotiations. This suggests that a great deal of common features can be isolated using the FSM description.

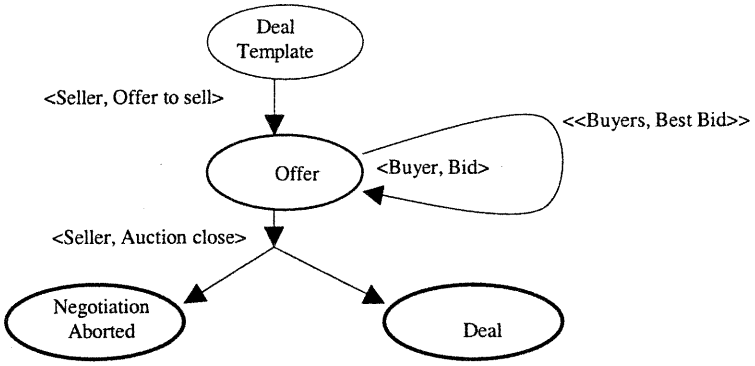

Figure 3.1: Finite State Machine description of an English auction [KF98a].

A FSM description alone is not sufficient to fully capture a negotiation process. It has to be complemented in order to answer the following questions: (1) what information

should be made available to the participants and what should be hidden, (2) what is the minimum starting bid, (3) what are the rules for closing the auction, etc.

In another study, Su et al. [SHH00] use a FSM to model bilateral bargaining. Here again, the states are the states of the negotiation and the transitions between states are labeled with "send" and "receive" primitives (e.g., send a call for proposal, receive an acceptance, etc.). The result looks similar to Kumar's and Feldman's FSMs, but the diagram produced is more complex and uses numerous states. According to Cass et al. [CLL+99], FSMs are used mainly to classify negotiations. They describe how the state of the negotiation changes in response to events; yet fail to describe the order of these events. This notation is purely a modeling notation and is not directly executable.

3.4.4 Statecharts

The FSM formalism alone cannot evidently capture a complete negotiation process. We propose to extend it by using Statechart diagrams [HG96] as adopted by the UML [RJB99]. Statecharts are well-established, widely used, and are semantically rich enough to formally describe and visualize various kinds of processes. As a further important feature, Statechart diagrams can be serialized in XMI [XMI98]. Finally, off-the-shelf simulation and analysis tools are available for Statecharts, such as *Statemate* [HLN+90], which will help to validate and render the descriptions being investigated. Below, we first discuss the OMG Negotiation Facility, which relies on a restricted form of Statecharts. Then, we present the fine-grained formalization which makes full use of the expressive power of Statecharts and which we adopted for our work.

3.4.4.1 The OMG Negotiation Facility

We present a proposal made by OSM [OMG99] in response to the RFP for a Negotiation Facility issued by the OMG. The proposal contains three negotiation models described using three Statechart diagrams, respectively. The three models aim to cover all the negotiable aspects of an e-commerce transaction and introduce two

new concepts: (1) negotiation through the introduction of motions, looking for consensus through voting, and (2) dealing with what comes after a deal is reached. For our work, only the bilateral negotiation model is relevant. It is defined as a collaborative process dealing with interactions between two participants (see Figure 3.2).

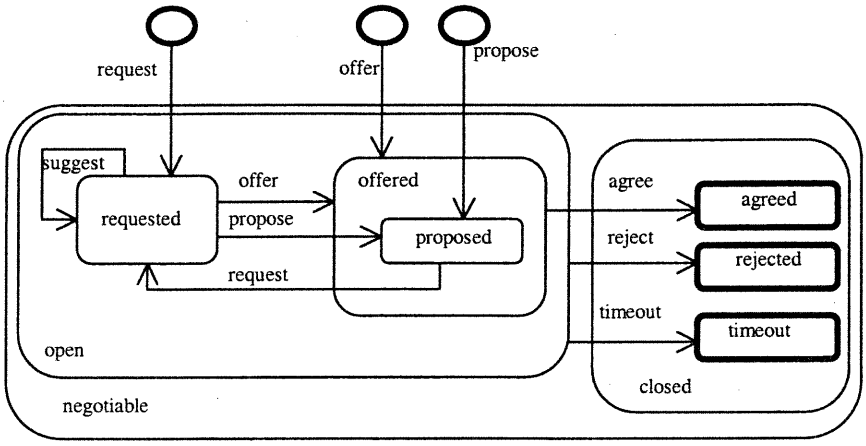

Figure 3.2: The OMG Bilateral Negotiation Model [OMG99].

The proposal considers a negotiation as a "give-and-take" process where two (or more) parties try to make concessions until they reach a deal. The OSM models are not well suited for describing auctions as they are intended to be generic at the cost of being precise. In a bidding situation, a bid can be thought of as an offer because it represents an engagement by the bidder to be honored if her bid is a winner. The OSM models, however, are based on the idea that a proposition is to be discussed until it becomes an offer, and then it can be agreed upon or not. Finally, since the OSM models are designed to capture all major negotiation types, the corresponding diagrams are hard to understand. Note that the models do not make full use of the Statechart formalism, in that, for instance, neither guard conditions nor actions appear in the diagrams.

3.4.4.2 Fine-grained Formalization

We propose to use the full expressive power of Statecharts, in particular the Event-Condition-Action strings provided for specifying transitions. We also suggest

describing each negotiation type separately. Figure 3.3 shows our Statechart description of an English auction. The main states are "Taking bids" and "Auction closed". When the auction is closed, it goes to the "Clearing" state where the auctioneer has to determine if there is a deal or not. The two possible final states are "Deal" and "No Deal". In the first case the seller and the buyer are notified. In the second case only the seller is notified. The transitions are labeled with the string *event[guard-condition(s)]action(s)*. So far, we have used Statecharts to describe some popular types of negotiations like the fixed-price sale, the Dutch auction, the bilateral bargaining, etc. We feel that the formalism is powerful enough to capture entire negotiation processes.

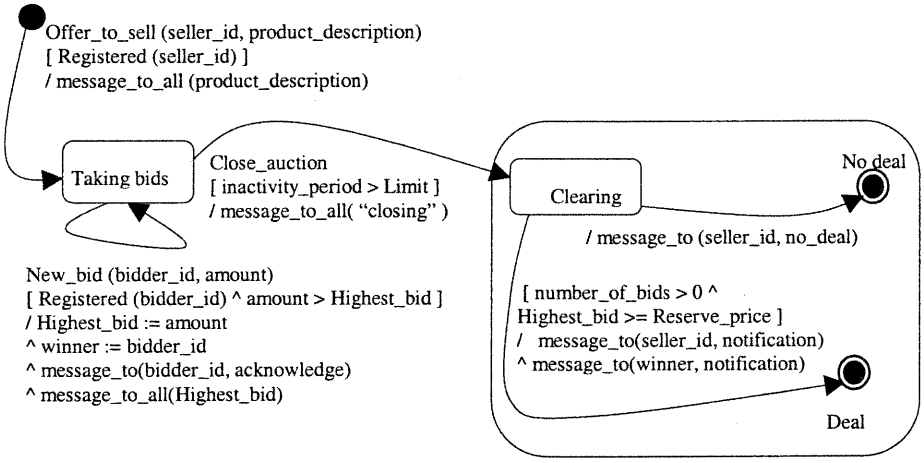

Figure 3.3: Statechart description of an English auction.

3.5 Comparison of Approaches

Table 1 summarizes our evaluation of description approaches. A + means that the description approach verifies the corresponding criterion, a - means that it does not, and a ? means that we could not confirm either case. The table shows that Natural Language is the big looser and that Statecharts satisfy all of our requirements. OMG's Negotiation Facility lacks clarity, mainly because the models try to capture all types of negotiations in single diagrams. Furthermore, the OMG descriptions do not reach the level of detail and completeness of the (full) Statechart approach. FSMs are highly rated, but they do not permit a complete description of negotiation processes

and are less popular than Statecharts. Agent Coordination Languages are even less popular and, to our knowledge, the descriptions cannot be serialized nor converted to other formalisms. We therefore adopted Statecharts as the formalism for describing negotiation rules.

	Formal Basis	Serial-ization	Visual-ization	Execut-ability	Popu-larity	Clarity	Complet e-ness	Convert -ibility
Natural language	-	-	-	-	+	-	+	-
Agent coordination language	+	?	+	+	-	+	+	?
Finite state machines	+	+	+	-	-	+	-	+
OMG negotiation facility	+	+	+	+	+	-	-	+
Statecharts	+	+	+	+	+	+	+	+

Table 3.1: Comparison of the five description approaches considered.

3.6 Conclusion

In this paper we demonstrated the need for a formal description of negotiation rules, and to this end, we reviewed and evaluated five different description approaches. The evaluation led us to suggest Statecharts as the best choice. They have a good formal basis and can be serialized, visualized, and executed. They are also well-established, are easy to understand, are complete, and can be converted to other formalisms. Our research relies on Statecharts to describe the individual negotiations that make up a combined negotiation. This will enable us to understand, implement and test the different negotiation types more thoroughly before using them in a combined negotiation. So far, we have used Statecharts to describe several popular negotiation types. Our future plans include modeling all the types described in [Ben00], and leveraging available Statechart tools for setting up the envisioned combined negotiation support system.

CHAPTER 4

COMBINED NEGOTIATIONS

Preface

In this chapter, we define Combined Negotiations (CNs) as a new negotiation type, and discuss the issues and difficulties they raise. A CN involves negotiating many interdependent items by engaging in different and independent negotiations.

We also detail the architecture of CONSENSUS, a support tool for conducting CNs. The tool helps the user to control and monitor the progress of the negotiations, making sure that the specified dependencies are respected, and applying user-defined negotiation strategies. The architecture is built around the following idea: *a workflow* that captures the sequencing and the control flow of the CN (*CN know-how*), *software agents* that carry out individual negotiations (*individual negotiation know-how*), strategy *rules* to help the agents decide what to do when there are many options to choose from (*negotiation strategy know how*), and *coordination rules* to manage the agents across several negotiations (*coordination know how*). An important challenge in this research is to allocate functionalities to these four parts in an optimal way.

Finally, we describe an implementation of CONSENSUS as a proof-of-concept prototype of the proposed architecture.

The relation of this chapter to the other chapters of the thesis is as follows: in Chapter 3, we evaluate mechanisms to describe the protocols of the negotiations that form a CN (i.e., the CN nodes), in Chapter 5 we show how GNP implements these negotiation protocols, and Chapter 6 complements this chapter by detailing the handling of negotiation strategies and coordination in CONSENSUS.

It is crucial that CONSENSUS not be confused with a negotiation server. GNP is a negotiation server but CONSENSUS is a Negotiation Support System (NSS). The

following NSS definition from [KL01] shall help making the distinction clear. "NSSs are designed to help and advise negotiators; they are used to structure and analyze the problem, elicit preferences and use them to conduct a utility function, determine feasible and efficient alternatives, visualize different aspects of the problem and the process, and facilitate communication." Although the definition does not fully apply to CONSENSUS, it applies partially as follows: CONSENSUS helps the user in conducting the CN, it structures the CN problem and visualizes it as a workflow, and it facilitates communication by automating it completely.

Our contribution to this chapter is as follows: we defined the CN model, identified the issues it generates, recognized the need for a support tool to help the user conduct it, designed the architecture for the support tool, prototyped and evaluated the concepts and technologies that were part of the architecture, and supervised the overall implementation and deployment process. The implementation of CONSENSUS was a team effort, heavily involving Hakim Alj and Mathieu Vézeau (co-authors of [BAV+00]) in all aspects of the prototyping, implementation, integration, and deployment process.

In the remainder of this chapter we present a modified version of the following publication:

Combined Negotiations in E-Commerce: Concepts and Architecture, by Morad Benyoucef, Hakim Alj, Mathieu Vézeau, and Rudolf K. Keller. *Electronic Commerce Research Journal,* 1(3):277-299, July 2001. Special issue on Theory and Application of Electronic Market Design. Baltzer Science Publishers.

Combined Negotiations in E-Commerce: Concepts and Architecture

Abstract. Combined Negotiations are a novel and general type of negotiation, in which the user is interested in many goods or services and consequently engages in many negotiations at the same time. The negotiations are independent of each other, whereas the goods or services are typically interdependent. Using currently available technology for electronic negotiations, the user conducts each negotiation separately, and has the burden of coordinating and reconciling them. The inherent complexity of combined negotiations in B2C as well as B2B e-commerce calls for software support.

In our research, we aim to devise a Combined Negotiation Support System (CNSS) to help the user conduct all the negotiations at the same time. The CNSS enables the user to control and monitor the progress of the negotiations, makes sure that the specified dependencies are respected, and applies user-defined strategy rules. We have designed such a CNSS, which we call CONSENSUS. The architecture of CONSENSUS relies on workflow technology, negotiating software agents, and rule engine technology. The originality of this architecture lies in the fact that the user of CONSENSUS models the combined negotiation at build time using a workflow that captures the sequencing of the individual negotiations and the dependencies between them. At runtime, software agents are assigned to individual negotiations, and they participate in the combined negotiation as actors in the workflow. The user can monitor the progress of the combined negotiation as a whole, and the progress of individual negotiations via dedicated graphical user interfaces. We rely on rule engine technology to enable the agents to use negotiation strategies.

The paper introduces combined negotiations with a usage scenario. Then, combined negotiations are detailed, along with the approach taken to cope with their complexity. Afterwards, we describe the functionality a CNSS should provide, and present the architecture of CONSENSUS, together with a discussion of the underlying concepts and technologies. Furthermore, we report on our prototype implementation of CONSENSUS and illustrate it with an example. A discussion of related and future work concludes the paper.

Keywords: electronic commerce, electronic negotiation, combined negotiation, auction, commitment, workflow, software agent, negotiation strategy, rule engine.

4.1 Introduction

Electronic negotiations (e-negotiations) are becoming an important research subject in the area of electronic commerce (e-commerce). The AMEC laboratory of MIT [AME00], for instance, puts e-negotiations at the center of its Consumer Buying Behavior (CBB) model for e-commerce [MGM99]. The model identifies six steps in an e-commerce transaction: (1) the need identification step in which the buyer is stimulated through product information, (2) the product brokering step in which information is retrieved to help the consumer determine what to buy, (3) the merchant brokering step in which the consumer determines who to buy from, (4) the negotiation step where the price and possibly other aspects of the deal are settled, (5) the purchase and delivery step, and finally (6) the product and service evaluation step where the product and service are evaluated by the consumer.

The most basic form of e-negotiation, as described by Kumar et al. [KF98b], is no negotiation at all (also called fixed-price sale) where the seller offers her goods or services through a catalogue at take-it-or-leave-it prices. This is the way most electronic retailers (e-tailers) operate. Auctions are a bit more complex, and they are at present the most visible type of e-negotiations on the Internet as conducted by eBay [Eba02], OnSale [Ons00], Yahoo [Yah02] and hundreds of other auction sites. A comprehensive description of the different auction types can be found in [Ago02]. E-negotiations can take an even more complex form called bargaining. This involves making proposals and counterproposals until an agreement is reached [San99b]. The Object Management Group (OMG) sees bargaining as bilateral and multi-lateral negotiation depending on whether there are two parties (one-to-one bargaining) or many parties (many-to-many bargaining) involved in the negotiation [OMG99]. Bargaining can become challenging if the object of the negotiation has more than one negotiable attribute (e.g., the price, the quality, the delivery date, etc.). In this case we talk of multi–attribute negotiations. A negotiation is said to be distributive (of win-lose nature) if a gain for one agent is necessarily a loss for the other agent [Ene02]. It is said to be integrative (of win-win nature) if the parties do not necessarily have

opposing interests, and they try to optimize different attributes. The buyer for instance can hope to pay a small price if she can live with a poorer quality and/or can stand a long delivery time. Combinatorial auctions [San99b] are another form of negotiations that involve making bids on combinations of goods or services, and one of the main challenges is for the auctioneer to determine the winning bid.

Jhingran [Jhi99] proposes a three-dimensional negotiation space (see Figure 4.1, solid lines). The first dimension is for the case where *multiple copies* of the same item (good or service) are available for negotiation, and the bids (i.e., offers) take the form of pairs (quantity, price-per-unit). The second dimension addresses the case where *multiple items* are subject to one negotiation. In this case, the participants make bids on combinations of these items. The third dimension is for *multiple attribute* negotiations. Jhingran's model obviously addresses only one aspect of a negotiation, which is the item being negotiated. A more complete classification of e-negotiations can be found in [Ene02]. The description identifies three more aspects of a negotiation: the participants (the "people" aspect), the type of the negotiation (the "process" aspect), and the criteria that can be used to evaluate the process (the "evaluation criteria" aspect).

We talk of a Combined Negotiation (CN) when a consumer is interested in many items, and consequently engages in many negotiations at the same time. The negotiations can be of any type (fixed-price sale, Dutch auction, bilateral bargaining, combinatorial auction, etc.). Each negotiation is for a separate bundle of copies, items, and attributes and thus corresponds to one point in the 3D negotiation space (see Figure 4.1, solid lines). The negotiations are in general totally independent of each other. The goods and services of the CN, however, are typically interdependent. To capture this new type of negotiation, where we have *multiple negotiations* going on at the same time, we introduced a fourth dimension in Jhingran's model (see Figure 4.1, dashed line).

As an example of a CN, let us take a vacation package consisting of three items: a transportation ticket, a hotel room, and a ski trip. The three items are obviously

interrelated since the consumer would have to travel to the location where the ski trip journey initiates (or at least near it) on the date of the trip (or before it). In addition to the places and dates, there can be other constraints and dependencies between the three individual items, as we will see later on. Let us suppose that the three items are negotiable (keeping in mind that a fixed-price sale is a special case of a negotiation) and that they can be negotiated on different negotiation servers. Let us suppose also that the negotiations practiced on each single server can be of different types. Clearly, the individual negotiations are independent of each other. Using currently available negotiation technology, the consumer would have to conduct each negotiation separately, and she would have the burden of coordinating and reconciling the various negotiations. It can happen for instance that the consumer makes a deal on a plane ticket and a hotel room, and then, while negotiating the ski trip, finds out (through bargaining for example) that she is missing out on a very interesting deal only because she arrives a few hours late.

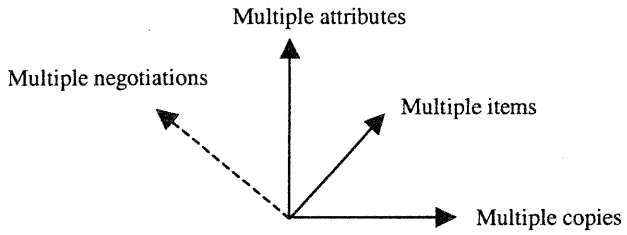

Figure 4.1: The four-dimensional negotiation space.

We therefore see a pressing need for a software system that supports the user in conducting all the negotiations at the same time, i.e., in carrying out CNs. We call such a system a Combined Negotiation Support System (CNSS). A CNSS is a tool that enables the user to track and monitor the progress of many negotiations efficiently and to respect all the constraints, dependencies and preferences of the given context. Moreover, a CNSS will support the user in taking decisions.

Before we go further, and to avoid any confusion, we shall first clarify the concept of "negotiation server" which will be heavily used in this paper. It is possible that many negotiations are created and conducted on one single server. These negotiations are

independent and therefore can run in parallel. An example of such a server is eBay [Eba02]. It is also possible that a server conducts only one negotiation. This is the case where a company makes a dedicated server available for its consumers to negotiate a particular item. In both cases, we will use the term "negotiation server". Another type of negotiation server accepts connections from users, takes a description of their needs and preferences, and then negotiates on their behalf [SHH00]. We will call this type a "proxy negotiation server".

The work presented in this paper is part of the TEM (Towards Electronic Marketplaces) project, a joint industry-university project started in 1999 involving researchers from economic science, software engineering, and operations research. The project addresses market design issues in respect to resource allocation and control and reward mechanisms, investigates open protocols for electronic marketplaces, and explores concepts and tools for e-negotiations. Two key tools being investigated in TEM are the CNSS described in this paper, and a generic negotiation server infrastructure, which we call GENESIS [BKL+00]. GENESIS is already functional and we are using instances of it to implement the servers involved in a CN. GENESIS is generic as it can support a great variety of negotiation types. It also offers the possibility to the participants to negotiate via a web browser or via software agents.

The contribution of this paper is twofold. First, we define the concept of CNs and present the problems they generate along with the approach taken to solve them. Second, we present an architecture for a CNSS we call CONSENSUS, which is based on workflow technology, negotiating software agents and rule engine technology. The originality of CONSENSUS stems from the fact that the user models her CN at build time using a workflow. The workflow itself captures the sequencing of the individual negotiations and the dependencies between them. At runtime, software agents are assigned to individual negotiations and participate in the CN as actors in the workflow. The user of CONSENSUS can track and monitor the progress of the CN via the Workflow Monitoring and Control Tool of a Workflow Management

System (WfMS). She can also monitor the work of the agents via an Agent Monitoring and Control Tool. The four-component model proposed by the Workflow Management Coalition (WfMC) [Wfm02] serves as the core of our CNSS architecture. We use Rule Engine technology to enable the agents to use negotiation strategy rules. These rules can be edited before the negotiations start and/or during the process of the negotiations. The formal description of negotiation rules[28] is also investigated, as it is necessary for CONSENSUS to transfer the negotiation rules from the corresponding negotiation servers to its Negotiation Rules Repository.

This paper[29] is organized as follows. In Section 2, we briefly describe a CN usage scenario. The concept of combined negotiation is detailed in Section 3, and in Section 4, we present our approach to solving the problems related to CNs. The architecture of CONSENSUS as well as an overview of its underlying concepts are covered in Section 5. Section 6 presents an example of a CN. Section 7 reports on the implementation of CONSENSUS, and Section 8 discusses related work. Finally, Section 9 concludes the paper.

4.2 Usage Scenario

Let us go back to our vacation package example. The consumer uses CONSENSUS to register, say, in one negotiation for the hotel room, two different negotiations for transportation, and one for the ski trip. CONSENSUS is then used to construct a workflow that models the CN (we will see the details later on). The workflow reflects the sequencing of the individual negotiations and the dependencies between them. The consumer might want, for example, to run all the four negotiations in parallel, or run all or some of them sequentially. The workflow also reflects the dependencies between individual negotiations such as the amount of money already committed and thus the remaining amount to be spent. After the consumer finishes modeling the CN, four software agents are instantiated and assigned to the four different negotiations. The agents are created according to the type of the negotiation practiced on the

[28] Negotiation rules mean the rules (i.e., the valid actions) of the game. A participant in the negotiation uses strategy rules to maximize her utility.

corresponding servers. The consumer can provide the agents with negotiation strategies such as: "if your bid is always beaten by the same opponent then be less aggressive in your bidding " or "if you have little chance of making a deal on the ski trip then don't commit yourself on the other two items of the package". Note that the first strategy applies to one individual negotiation whereas the second one applies to the CN as a whole. The negotiation strategies can be communicated to the agents before the negotiation starts or during the course of the negotiation. This scenario is an example of B2C e-commerce. In case one or more items in the package are offered by consumers (i.e., a rare ticket to a rock concert auctioned on an auction site), we would face a C2C transaction.

CONSENSUS can also be used at the B2B level. A travel agency, for instance, can use CONSENSUS to negotiate travel packages on behalf of its clients. The more items there are to be negotiated and the more providers of such items there are, the more there is a need for a support tool. We might imagine different means of transportation (by air, by bus, etc.), different types of accommodations (hotel, motel, pension, etc.), and different recreational activities (ski trip, camping, hiking, etc.). We might also imagine that there are many providers of such services and that each provider might practice a different type of negotiation. As another example at the B2B level, consider a company that wants to import some merchandise from a foreign country. The attributes and issues to be negotiated are: the price as well as other attributes of the merchandise, the transportation, the insurance, the customs, etc. These examples suggest that at the B2B level, CNs may become highly complex, and that automated support by a CNSS is even more important.

CONSENSUS offers the possibility to find the products and their providers (steps 2 and 3 of the CBB model) along with complete information to help the user decide which products to negotiate and where to negotiate them. The negotiation types practiced by the corresponding servers should be known to the user as it is an important factor in choosing one provider over another. She might for example prefer

---

[29] This paper is a revised and extended version of [BK00b].

a bargaining type negotiation to an English auction if she is a good haggler. Once the choice is made, the CN can be modeled. A high-level specification tool is provided by CONSENSUS to specify the workflow and an English-like syntax is used to specify the negotiation strategies. When the modeling phase is over, the CN is launched by running an instance of the workflow, which in turn launches the software agents. CONSENSUS provides the user with the possibility to track the individual negotiations as well as the CN as a whole. The user can also edit negotiation strategies at run time.

We see CONSENSUS as a support tool, where the automation provided by the workflow engine (to run the CN) and the software agents (to run the individual negotiations) should allow the user to intervene in the important aspects of the negotiation whenever a need should arise. Our CNSS is not meant to replace the user, but to provide her with a powerful tool.

4.3 Issues in Combined Negotiations

In this section, we address seven important issues related to CNs. We first define CN failure, and then present the notions of AND-Negotiation and OR-Negotiation. Thereafter, we discuss the negotiation types covered by a CN. Then, constraints in a CN are explained. We then present the notion of commitment in negotiation protocols followed by a comparison of the CN and the synchronized auction types. Finally, we classify the information involved in a CN.

4.3.1 Failure in Combined Negotiations

When we engage in a negotiation, there is no guarantee that we will end up as a winner, and even if we do win, there is no guarantee that we will make a good deal by paying the real value of the good (this last case is referred to as the *winner's curse* meaning that the winner has possibly over-evaluated the item and ends up paying more than the item is worth).

In a CN the risks for the user are even higher because she has to negotiate several items that form a package. Whether the individual negotiations are conducted in

parallel or in sequence, there is always the risk that some individual negotiations will be successful and some will not be. Since the user wants the whole package or nothing, she might want to break (if allowed) the commitments she made in the successful negotiations. Breaking a commitment evidently has a price. We talk of failure (also referred to as exposure) in a CN, when we need two items A and B, engage in negotiations on both items, and end up winning on A but loosing on B.

4.3.2 AND-Negotiation and OR-Negotiation

The package that is the object of the CN is in general made up of many items. If we engage in many negotiations for the same item we call these OR-Negotiations. In the case of a vacation package, for example, we can engage in more than one negotiation for the plane ticket. Negotiations for different items that make up the package are called AND-Negotiations. We might, for example, engage in one negotiation for the plane ticket (possibly an OR-Negotiation), another one for the hotel room, etc.

OR-Negotiations. OR-Negotiations will usually run in parallel. This means that, in order to save time and also to maximize the chances of a good deal, the negotiations for one item are started at the same time. This will enable the user to see the progress of the negotiations and choose to bid (make offers, counteroffers) in the most promising one. We may have to ensure that no more than one commitment (bid, offer, etc.) is made at the same time, that is, commitments must be mutually exclusive. If we do not take this precaution, we might end up making more than one successful deal on the same item. An alternative scenario would be to run OR-Negotiations sequentially. Whenever one fails (the bids are too high, the negotiation ended, etc.), we start another one for the same item.

AND-Negotiations. Here again the user has the choice of running her individual negotiations in parallel or sequentially. Running them (or some of them) in parallel will be more interesting for the user. She can make decisions in one negotiation based on what is happening in the other ones. The type of the negotiation can also make it appealing to run the AND-Negotiations in parallel, as we will see in the next subsection.

4.3.3 Negotiation Types covered by a Combined Negotiation

The individual negotiations for the different items in the package can be of any type. The consumer might engage in a bargaining type negotiation for the ski trip, an English auction for the plane ticket, and a Dutch auction for the hotel room. Each negotiation type evidently will require different rules and strategies. Therefore, the CN depends on the types of the individual negotiations. If the consumer is interested in an item that is offered in a sealed-bid auction that ends in 10 hours, she might for example try to finalize deals on the other items in the package and use the remaining amount of money to make a bid in the sealed-bid auction.

The fact that more than one attribute of an item is negotiable in general complicates the dependencies among the individual negotiations. Let us go back to the vacation package and suppose that it is made up of three items: a plane ticket (price, date, week-day/week-end, first-class/second-class, destination, etc.), a hotel room (price, date, type, location, etc.), and a ski trip (price, date, location, etc.). The outcome of one negotiation will be crucial in the other ones (the date of the flight from the first negotiation will be a necessary input to the other two negotiations).

These examples suggest that multi-attribute negotiations, possibly of different types, make CNs flexible. Yet, such negotiations may incur complex dependencies among the items that form the CN package and thus warrant software support.

4.3.4 Constraints in a Combined Negotiation

Let us consider a CN for three different items i1, i2 and i3. The individual negotiations are N1, N2 and N3. Suppose that the attributes for each item are the price, the date, and the place. We classify CN constraints as intrinsic or procedural.

Intrinsic constraints: These constraints concern the dependencies between the items, copies, and attributes of the CN. They may involve just one single individual negotiation, such as:

- price1 * number-of-copies <= threshold.

- date1 in RANGE.
- place1 in (X, Y, Z).

Other constraints may involve more than one of the individual negotiations forming the CN. Examples of such constraints include:

- price to pay for the package <= threshold.
- date3 = function(date1, date2).

Procedural constraints: These constraints indicate the sequencing in time of the individual negotiations and the control flow between them. Examples of such constraints include:

- Sequential: N2 is launched after N1 is finished.
- Parallel: N1 and N2 are launched at the same time.
- Choice: depending on a condition, either N1 is launched or N2 is launched.
- Wait for: N3 waits for N1 and N2 to finish or waits for either one to finish (we suppose that N1 and N2 both start before N3 and that N3 is aware of the status of N1 and N2).
- Repeat: repeat N1 until a condition is met.

4.3.5 Commitment in Negotiation Protocols

A further issue of importance in CNs are commitments in negotiation protocols. It is obvious that, given the possibility of breaking a commitment (even at a certain cost to the user), the CN will be more flexible and the role of the CNSS even more important. According to Sandholm and Lesser [SL95], commitment means that one agent (human or software agent) binds itself to a potential contract while waiting for the other agent to either accept or reject its offer. If the other party accepts, both parties are bound to the commitment. When accepting, the second party is sure that the contract will be made, but the first party has to commit before it is sure. As an example, let us consider an English auction with the possibility to break a

commitment. If your bid is the winner (a commitment from you) and you decide to leave the auction (break the commitment), then you will have to pay at least the difference between your bid and the second highest bid (or be subject to another penalty agreed upon in advance).

4.3.6 Combined Negotiations versus Synchronized Auctions

A CN bears some resemblance to a well-known auction type called the synchronized auction, and for that reason, we would like to point out the differences and similarities between the two. The following definition is taken from [Ene02]. In a synchronized (also called simultaneous or parallel) auction there are n items. A participant can make m distinct bids on m distinct items (m not greater than n). The m bids are made simultaneously. The auction usually runs multiple rounds of sealed bids, announcing the bids after each round. The Federal Communication Commission (FCC) of the U.S. uses such a format to auction licenses for Personal Communication Services (PCS). Note that a variation of the synchronized auction is the so-called combinatorial (or bundled) auction [San99b], which allows the participant to make a single bid for m possibly distinct items (m not greater than n).

The similarity between a CN and a synchronized auction is the fact that the user can make many distinct bids (offers or counteroffers) on many distinct items, and that the items might be interrelated. However, a CN may involve many different individual auctions and/or negotiations, whereas the synchronized auction is one single auction with a set of known rules. Moreover, a CN is not synchronized. A CN should not be confused with a combinatorial auction either. In a CN, the user cannot make a single bid on all the items of the package.

4.3.7 Information Involved in a Combined Negotiation

The negotiation process is an exchange of information between the participants and the negotiation server. It "consists of a number of decision-making episodes, each of which is characterized by evaluating an offer, determining strategies and generating a counteroffer" [WZK00]. In order to evaluate an offer, the participant needs all the

information she can get from the negotiation server. Based on that information, she can choose a negotiation strategy that helps her shape a counteroffer in response to the offer made by the server. We define a negotiation strategy as the way in which a human negotiator would react in her best interest in a given situation and given the information that is available to her. Two examples of negotiation strategies in an English auction are: "when the highest bid reaches 1000 dollars, then make a bid by doubling your bid increment" and "if the frequency of bids is low, then be less aggressive in your bidding." Here "the highest bid" is equivalent to the "offer" coming from the server, and "make a bid" is equivalent to the counteroffer to be made by the participant. How much the participant bids is decided by applying negotiation strategies that rely on information that we classify as follows:

- Internal versus external: the information can be internal to the CN (the total amount the user is willing to pay, her reserve price, etc.) or external (the current quote or highest bid, the identity of the other participants (if available), the profile of the other participants (if available), etc.).
- Individual versus combined: it can be related to an individual negotiation (the actual quote, the time remaining, etc.) or to the CN at large (the number of negotiations where the consumer is leading, the total amount of money already committed, etc.).
- Static versus dynamic: it can be static (the total amount the user is willing to pay, etc.) or dynamic (the frequency of bids, the average bid increment, etc.).

4.4 Tool Support for Combined Negotiations

We are aiming at a tool (CONSENSUS) that helps manage and possibly minimize the risks that the consumer faces in a CN. We see the tool as a support system for the consumer and not as a system that can replace her. Furthermore, note that CONSENSUS is quite different in scope from Negotiation Support Systems (NSS) [LK99]. The main idea behind these latter systems is to help the user evaluate offers made by her opponents, and evaluate her counteroffers before she submits them to

her opponents. All this happens in a bargaining type negotiation. This does not preclude, of course, the use of NSSs on top of CONSENSUS, in order to assist the agents in their evaluation of offers and counteroffers.

The functionality of CONSENSUS can be summarized in the following four points:

- Model the CN using a workflow definition formalism to specify its intrinsic and procedural constraints. We realize that in general there may be a need to change both the intrinsic and the procedural CN constraints dynamically, that is, during execution of CONSENSUS. Accordingly, CONSENSUS should support such dynamism.
- Use software agents for automation. Software agents are assigned to individual negotiations.
- Use a WfMS to coordinate, track, and monitor the work of the negotiating software agents.
- Use negotiation strategy rules to make the agents more autonomous. The user should have the possibility to enter strategy rules and edit them at runtime.

As seen earlier, there is a CN failure when we need two items A and B, we engage in negotiations on both items, and end up winning on A but loosing on B. We propose the following three solutions to address CN failure:

- Provide negotiation rules that allow the user to break a commitment. In this case the user backs off and breaks her commitment on item A and (eventually) pays a price for that. This is a CN failure resolution solution.
- Engage in many OR-Negotiations so that whenever we loose on item B we engage (or resume) a negotiation on item B on another server. This, of course, is not a guarantee against exposure. This is a CN failure prevention solution.
- Use strategies to minimize the risk of exposure. An example would be: if you notice that the competition is high on item B (and that you have little chance of winning), then do not make a commitment on item A. Here, once more, we

see the need for coordination between individual negotiations. This is again a CN failure prevention solution.

Finally, a commitment can be different from one negotiation type to another. In an English auction, to bid means that you make a commitment to buy in case you stay leader until the closure of the auction. In a Dutch auction, however, by bidding you make a commitment that is binding and final. In a "bilateral bargaining" type negotiation, offers and counteroffers are exchanged between the user and the negotiation server until an agreement is reached or until the negotiation is aborted. Offers become "binding" only when they are "final" (i.e., if accepted they must be honored). CONSENSUS will take into account the different types of commitments. If the rules of the negotiation allow for commitments to be broken, then this fact should be reflected in the rules describing the negotiation and in the workflow that models the CN.

4.5 Architecture of CONSENSUS and Underlying Concepts

In this section, we first describe the architecture of CONSENSUS. Then, we give a brief overview of the concepts and technologies it is based upon, that is, the formalism for describing negotiation rules, software agents, workflow management, as well as negotiation strategies and rule engines.

4.5.1 The Architecture

Figure 4.2 shows the architecture of CONSENSUS. The *Product/Merchant Brokering System* (responsible for phases 2 and 3 of the CBB model) is used by the consumer to select the products and the providers she is willing to negotiate with. The system will return a list of candidate negotiations, and the user selects the ones she is interested in. This list is passed on to the *Agent Factory*. The Negotiation Rules (NR) are then downloaded from the corresponding negotiation servers. We are using XMI [XMI98] as a vehicle for such a transfer. The rules are stored in the *NR Repository* and are used by the *Agent Factory* to instantiate the *Agents* that will be responsible for the individual negotiations. Each agent will be instantiated according to the

negotiation type practiced by the negotiation server assigned to it. After they are instantiated, the agents connect to and register on the corresponding servers. The consumer uses the *CN Specification Tool* (a GUI tool) to specify the CN. A constraint language will be used to represent, manipulate and store the constraints of the CN. The constraints are stored in the *CN Repository*. Another part of the specification deals with negotiation Strategy Rules (SR). These go into the *SR Repository*. When the CN specification task is completed, the workflow modeling the CN is generated and stored in the *Workflow Repository*. The consumer can use the *Workflow Definition Tool* to see and debug the resulting workflow. This ends the modeling part. At runtime, an instance of the workflow is started.

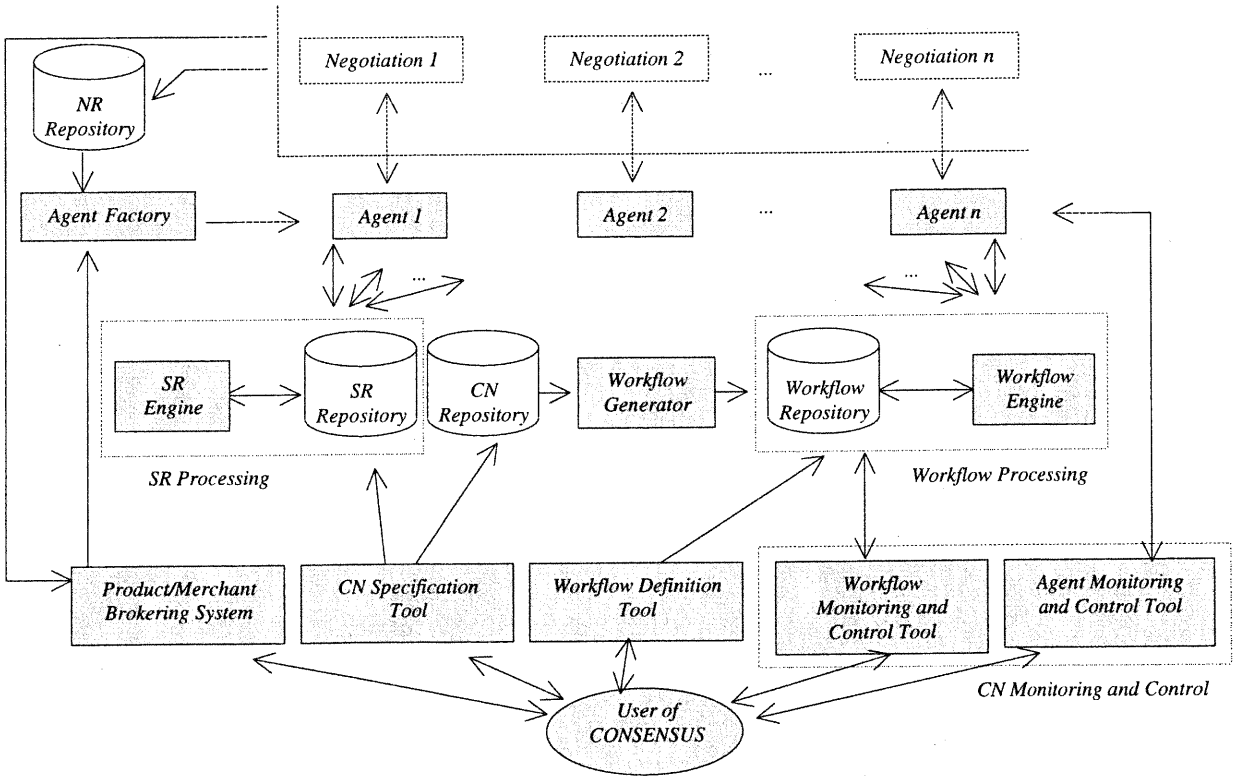

Figure 4.2: Architecture of CONSENSUS.

The workflow can be monitored via the *Workflow Monitoring and Control Tool*. The *Workflow Engine* executes the workflow by dispatching work to the agents (which

are actors in the workflow) and enforces the sequencing of the activities in the workflow. The user can monitor the work of the agents (what is specific to individual negotiations) using the *Agent Monitoring and Control Tool*. Strategy rules can be entered and/or edited at run-time. Each *agent* instantiates an *SR Engine* to make inferences based on the rules found in the *SR Repository*. Note that *Negotiation 1, Negotiation 2, ... Negotiation n* are the negotiations in which the agents participate. They can be conducted on one single negotiation server or on different ones as long as they are independent of each other.

4.5.2 Formal Description of Negotiation Rules

The transfer of the negotiation rules from the servers into the *NR Repository* of CONSENSUS as well as the instantiation of the agents by the *Agent Factory* calls for a formal description of these rules. We strongly believe that the rules describing the negotiation are as important as, or perhaps even more, than the other information describing the good or service that is the object of the negotiation. For that, they should be known to the participants (human or software) before they engage in the negotiation. We agree with Wurman et al. [WWW98] that "implementing auction mechanisms in a larger context will require tools to aid agents in finding appropriate auctions, inform them about auction rules, and perform many other market facilitation functions". Therefore there is a need for a mechanism to formally describe the rules governing a negotiation, visualize the description when necessary, and serialize it in order to transfer it over the network. This mechanism should also make it possible to separate the description of the negotiation process from the other parts of the negotiation software, for the purpose of efficiency, reuse, and ease of testing. A review and an evaluation of the methods used to describe negotiations can be found in [CLL+99, BK00a].

Inspired by Kumar and Feldman's Finite State Machine approach [KF98b], we have adopted the UML's statechart diagrams [RJB99] to describe the negotiation processes. An important feature of statechart diagrams is the possibility to be serialized in XML [XML98] and XMI [XMI98]. Moreover, simulation and analysis

tools are available for statecharts, such as Statemate [HLN+90], which help validate and render the descriptions being investigated. Figure 4.3 shows our statechart description of an English auction. The main states of the English auction are "Taking bids" and "Auction closed". When the auction is closed, it goes to the "Clearing" state where the auctioneer has to determine if there is a deal or not. The two possible final states are "Deal" and "No Deal." In the first case the seller and the buyer are notified. In the second case only the seller is notified. The transitions are labeled with the string *event[guard-condition(s)]/action(s).*

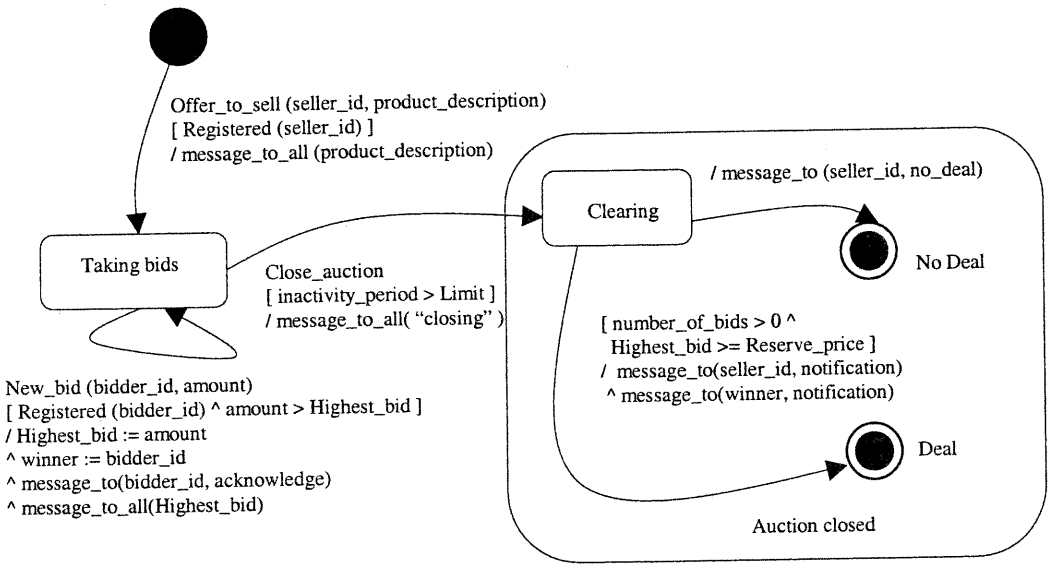

Figure 4.3: Statechart diagram of an English auction.

4.5.3 Negotiating Software Agents

The agents in CONSENSUS are instantiated by the *Agent Factory* and will act as participants in the workflow. Intelligent agents can be defined as software entities that execute functionalities in an autonomous, proactive, social, and adaptive fashion. These functionalities include searching, comparing, learning, negotiating, and collaborating [Jon99]. This makes software agents particularly useful for the information-rich and process-rich environment of e-commerce [MGM98]. They are well suited for information filtering and retrieval, personalized evaluation, complex

coordination, and time-based interaction. This last feature is vital to e-negotiations. The agents in CONSENSUS are autonomous and proactive, that is, they do not require the intervention of the user as long as they are provided with strategy rules. Furthermore, they are intelligent by making inferences using the *SR Engine*. Recall that the *SR Engine* processes strategy rules, in order to decide, for instance, whether or not to make an offer and on the amount of the offer. Our agents reside on the machine that runs CONSENSUS. In contrast, the agents of the AMEC laboratory, for instance, can be mobile, which enables them to go from site to site to negotiate deals on behalf of their creator [CM96]. Our agents as well as those of the AMEC laboratory are used in the negotiation step (Step 4 of the CBB model), whereas several commercial products use agents in the preceding steps, such as Amazon [Ama02] in the need identification step (Step 1), PersonaLogic [Per00] in the product brokering step (Step 2), and Bargain Finder [Bar00] in the merchant brokering step (Step 3).

Parkes et al. distinguish autonomous and semi-autonomous agents: "a fully autonomous agent requires a complete set of preferences in order to represent the user correctly in all situations that it might encounter," and "a semi autonomous agent will bid on behalf of the user when it has enough knowledge to proceed, and query the user when its best action is ill-defined given the current information" [PUF99]. A further distinction is being made between a reservation-price agent and a progressive-price agent. The first type is an autonomous agent that places bids up to the value of a fixed reservation price. The second type, on the other hand, is initialized with a lower bound and an optional upper bound on the true reservation prices of the user [PUF99]. We intend to support in CONSENSUS semi-autonomous and fully autonomous agents, as well as the two facets of price agents.

4.5.4 Workflow Management

The architecture of CONSENSUS is based on three WfMS modules: the *Workflow Definition Tool*, the *Workflow Engine,* and the *Workflow Monitoring and Control Tool*. These modules correspond to three of the four components suggested by the

WfMC [Wfm02]: (1) The Process Definition Tool which is used to enter the workflow into the computer, (2) the Workflow Engine which executes and tracks the workflow, (3) the Administration and Monitoring Tool used to administer and track the status of the workflow, and (4) the Workflow Client Application through which the participants interact with the workflow. We do not introduce a Client Application component, since in our solution the participants are software agents.

We rely on workflow technology because it provides support in three broad functional areas [Wor98]: (1) workflow definition: capturing the definition of the business process; (2) workflow execution: managing the execution of the workflow processes in an operational environment, sequencing the various activities to be performed and; (3) workflow monitoring: monitoring the status of workflow processes and dynamically configuring the runtime controller. Note that these functional areas are reflected in the WfMC components and the CONSENSUS modules mentioned above.

In the architecture of CONSENSUS, the workflow captures the logic of the CN (i.e., its intrinsic and procedural constraints), whereas the agents capture the logic of the individual negotiations. The agents, by participating in the workflow, can share information and cooperate in conducting the CN. Using a workflow also makes it easier for the user to track and monitor the progress of the CN at runtime. Furthermore, the user can adjust certain intrinsic and procedural constraints at runtime. Examples include adjusting the total price she is willing to pay at runtime, or changing the range of acceptable dates for her flight. We are also investigating the possibility to generate a workflow from a higher-level specification so that the user need not learn how to model a workflow in order to use CONSENSUS.

4.5.5 Negotiation Strategies and Rules Engines

According to Wong at al. [WZK00], most current e-commerce systems use predefined and non-adaptive negotiation strategies in the generation of offers and counteroffers during the course of the negotiation. Commercial auction sites such as eBay, for instance, still require that consumers manage their own negotiation

strategies over an extended period of time [MGM99]. There is however a possibility to relieve the user from managing the negotiation. EBay, for instance, offers the possibility of "proxy-bidding" which is actually a simple agent-negotiation that uses a straightforward strategy: bid until you reach a certain amount, by going up each time with a certain increment (called the bid-increment). As another example, the buying agents in the well-known KASBAH system [CM96] can chose between three negotiation strategies: anxious, cool-headed and frugal, corresponding to linear, quadratic, and exponential functions, respectively, for increasing their bid for an item over time.

In the architecture of CONSENSUS, we use rule engine technology to represent and exploit negotiation strategy rules. A rule such as: "if the bidding gets too high, then abandon the negotiation" can be nicely coded as a declarative statement. According to McClintock et al. [MB00], "rules are declarative statements that drive activity in a software application by describing the action or actions to take when a specified set of conditions is met". The rules can be coded as stand-alone atomic units, separate from and independent of the rest of the application logic. This makes the rules easier to develop and maintain. Moreover, rule engines have special languages for writing rules.

A rule engine is a software component designed to evaluate and execute rules [MB00]. Rule engines are already been used in consumer profiling (also referred to as e-commerce personalization). The idea is to use collected information about consumers visiting a web site in order to trigger rules that tailor the content of a provider's site to the profiles of the consumers. The information can be collected during all steps of the CBB model (see Section 1) and, in the above example, it is used in Step 1 of the model.

A similar declarative approach is taken by Su et al. [SHH00]: "Each strategy is expressed in terms of an Event-Trigger-Rule (ETR); which specifies the condition to be checked and the actions to be taken by the negotiation server. Rules are activated upon the occurrence of specific events during the negotiation." Instead of a rule

engine, they use an ETR server [LS98]. The server manages events and triggers rules that are relevant to the posted event.

4.6 An Example of a Combined Negotiation

In this section, we illustrate how a CN is modeled using a workflow and strategy rules. Going back to the vacation package example, we start with a simple CN with no parallel negotiations. The CN is for a vacation package consisting of two items: a hotel room and a transportation ticket. Depending on the price the consumer will pay for the hotel, she either negotiates a plane ticket or a boat ticket (we suppose that it is cheaper to travel by boat). We use IBM's MQSeries WfMS [IBM02] build-time client for modeling.

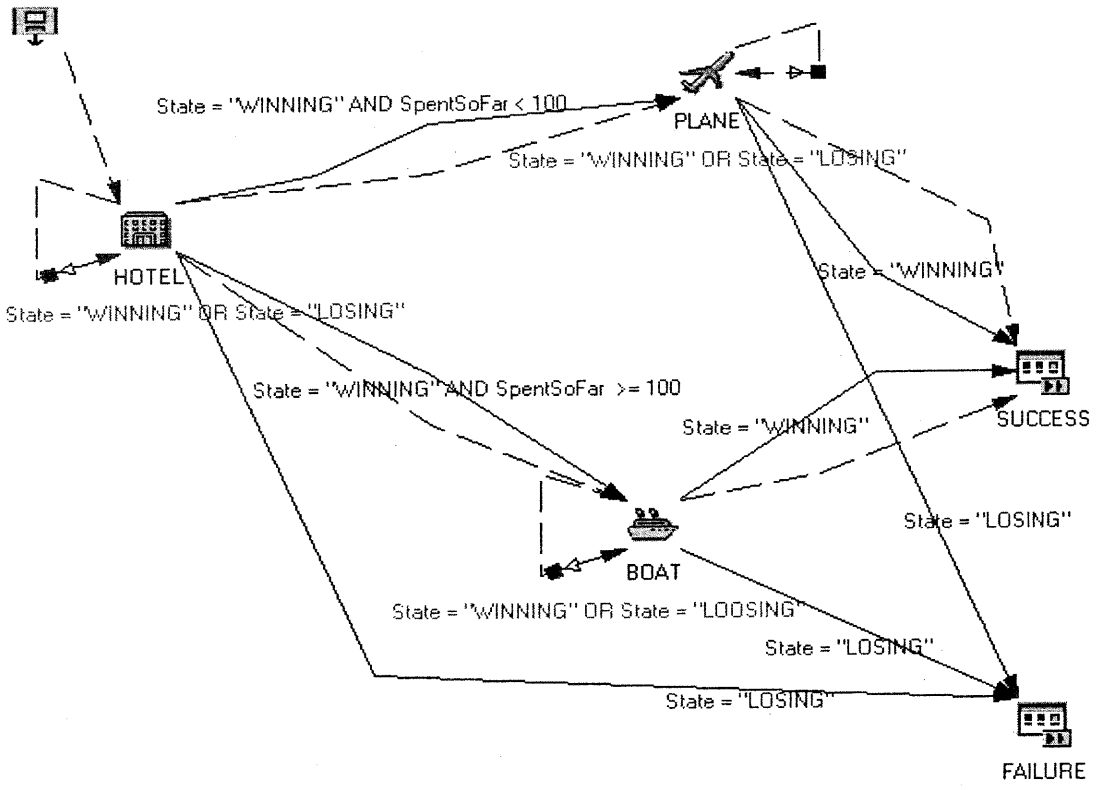

Figure 4.4: Workflow model 1 for the travel package.

Figure 4.4 presents the resulting workflow. There are 5 nodes that represent activities. Three activities represent three negotiations: the hotel room, the plane ticket, and the

boat ticket. These activities will be carried out by software agents (note that in other domains, workflow activities are typically assigned to human agents). The two other activities deal with the success and failure of the CN, and specific programs written for that purpose will carry them out. The five activities are connected by control connectors (solid arrows) and data connectors (dashed arrows). Transition conditions are associated to control connectors. Exit conditions are associated to the three negotiation activities, which means that the activities are to be repeated until these conditions are verified. The workflow shown in the figure states the following: "Start negotiating the hotel room. Keep negotiating until you win or you lose. If you win by paying less than 100 units then start negotiating a plane ticket. If you win by more than 100 units then negotiate a boat ticket. If you lose, stop the CN."

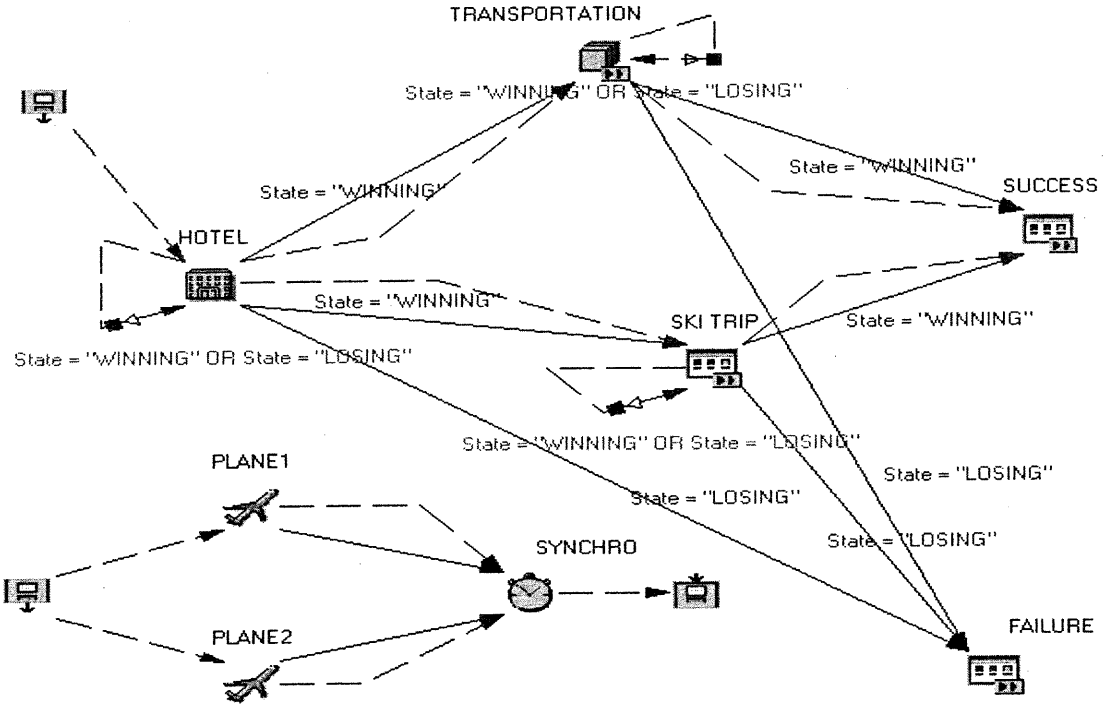

Figure 4.5: Workflow model 2 for the travel package.

Let us add a third item to the vacation package (a ski trip). Let us also choose to travel only by plane and introduce parallel negotiations. In Figure 4.5, the modeling is the same as before except for the node "TRANSPORTATION" which, in this case, is

not an activity but a block of activities. A block represents a sub-process (sub-workflow). The block "TRANSPORTATION" in the figure is equivalent to the sub-workflow on the lower left corner of the figure. The main workflow states: "start negotiating a hotel room and if you win, then start the negotiations for the transportation and the ski trip at the same time. The sub-workflow "TRANSPORTATION" and the activity "SKI TRIP" will be started at the same time. That gives three negotiations going on at the same time (the sub-workflow will launch two parallel negotiations for plane tickets). Obviously, we have to be careful not to win two plane tickets. To solve this problem, we intend to use strategy rules. An obvious rule would be: "never make more than one commitment for a plane ticket at the same time." Another rule could be: "only bid in the negotiation with the smaller bid." Since the hotel room is already booked, the place and date attributes are also fixed for the other negotiations. The total amount of money to spend for the package will have to be respected. The sum of the bids in the parallel negotiations should not exceed that amount. The activity "SYNCHRO" in the sub-workflow is used to synchronize the two negotiations for the plane ticket, taking control after the two agents make a pulse.

The workflow by itself in not sufficient to model the CN. To complete it, we need strategy rules such as the one in Figure 4.6. The notation used is that of ILOG JRules [ILO02] which is based on an English-like syntax. Rules have a "WHEN part" which specifies the conditions that must be met in order for the "THEN part" to be executed. The rule in Figure 4.6 applies to an English auction, and it states: if the participant is not leading (i.e., her bid is not the highest bid) and her reserve-price is about to be reached (i.e., the difference between the reserve-price and the highest bid is less than a certain limit) and she can reduce her increment while still respecting the minimum permitted, then she should reduce the increment by half.

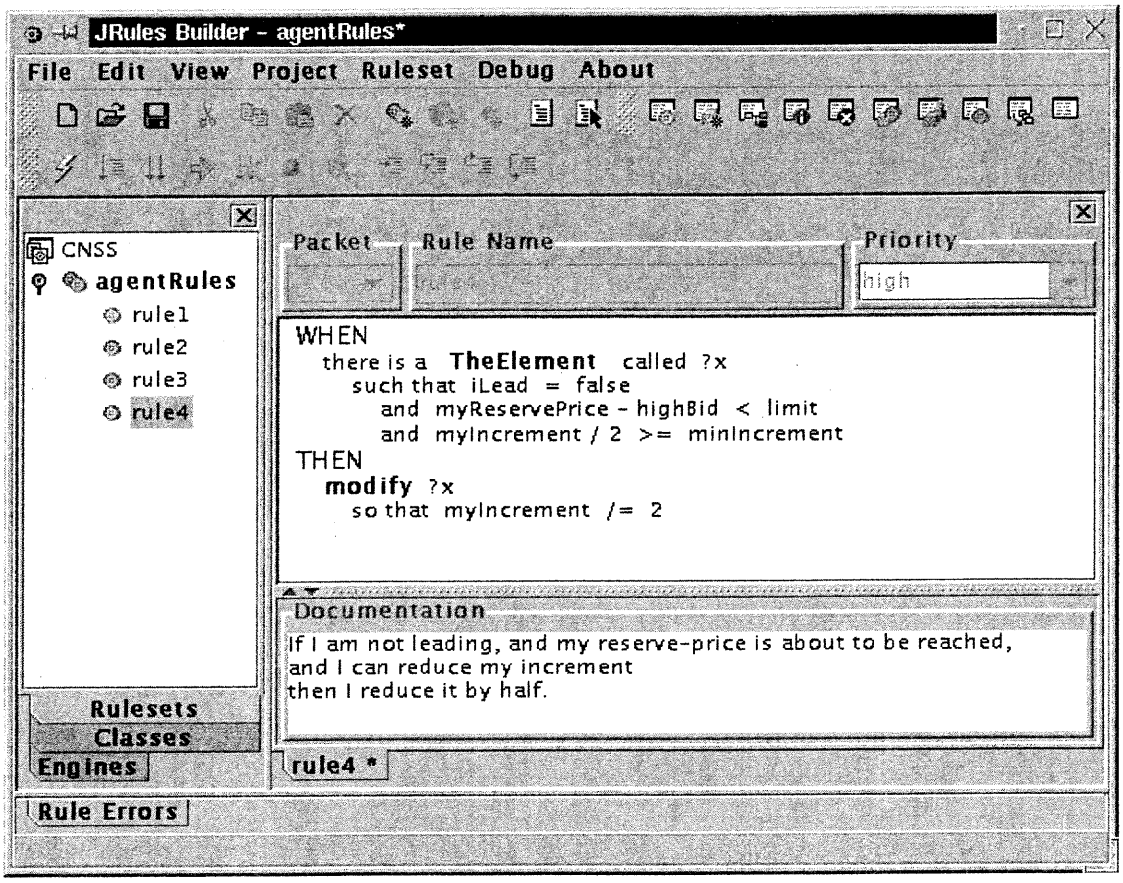

Figure 4.6: A sample strategy rule.

4.7 Implementation of CONSENSUS

Part of the presented CONSENSUS architecture has already been implemented as CONSENSUS version 0.1. This version includes a WfMS, software agents and an Agent Monitoring Tool. Instances of our negotiation server GENESIS are used by CONSENSUS to conduct individual negotiations. Only one negotiation type is supported in this version (the English auction). It is easy to implement new negotiation types on GENESIS since all there is to do is to write a new script describing the new negotiation type (called the auctioneer in GENESIS terminology). The agents function in pulses (also called episodes). A pulse is an atomic action by the agents, which includes getting the information from the server, deciding what to do, and finally taking action. The concept of pulse adds transparency to the negotiation process. The WfMS gets the control back after each pulse by the agents.

The exchange of information between the agents and the servers is done via XML documents (the order, the quote, the adjudication, etc.). We use the IBM MQSeries WfMS for this version and rely on its build time client for modeling and on its run time client for executing instances of the workflow model and for monitoring. There is no use of negotiation strategies in this version, and consequently, the second action of the pulse (deciding) will be performed by the user.

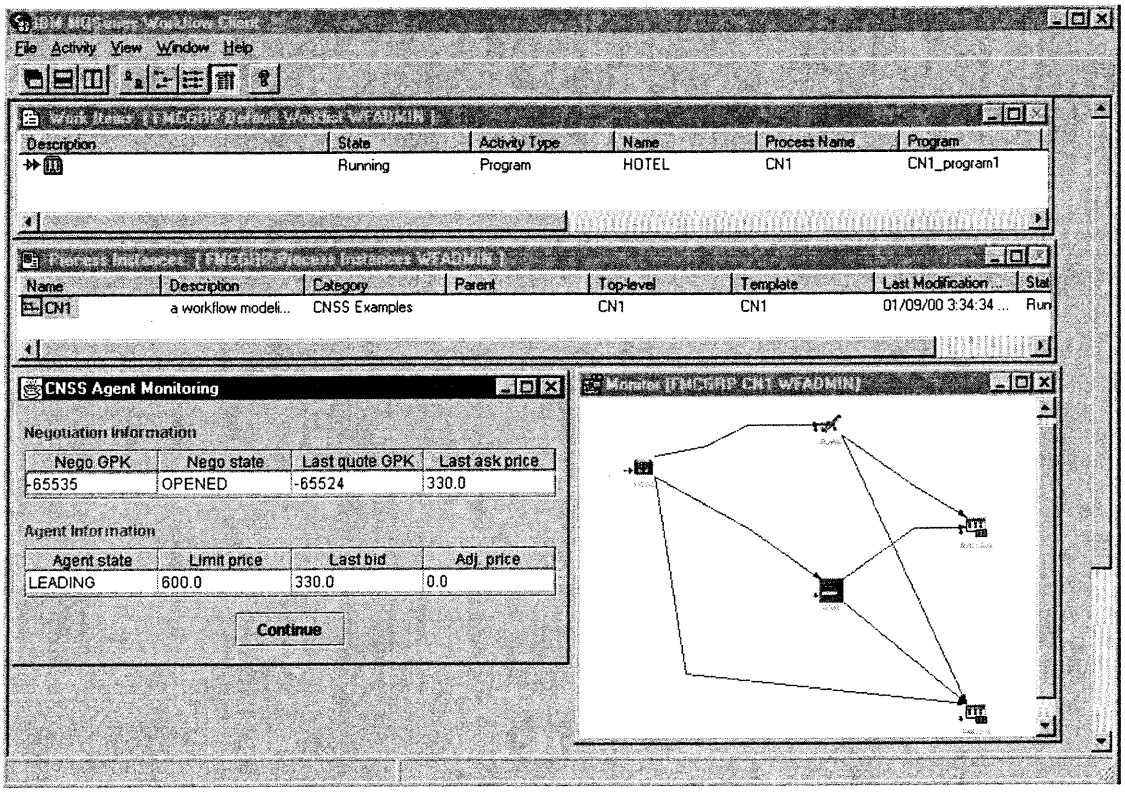

Figure 4.7: Screenshot of CONSENSUS version 0.1.

In a standalone fashion, we have incepted integrating rule engine technology into our negotiating agents. As a rule engine, we are using ILOG's JRules. The integration with CONSENSUS will come later, once we are satisfied with the results of using negotiation strategies in individual negotiations. Since strategies for whole CNs are much more complex than those for individual negotiations, we will deal with them in later versions.

In Figure 4.7, we present a screenshot of CONSENSUS version 0.1. The main window is that of the IBM MQSeries WfMS runtime client. The window at the top shows the work items (In case of manual processing, the human will have to consult this window to know what to do. In our case, the agents are activated automatically). The second window shows the instance of the workflow that is running. The third window (bottom left) shows the Agent Monitoring and Control Tool displaying information about one of the individual negotiations that is going on. The last window (bottom right) is the monitoring window of the IBM MQSeries WfMS.

CONSENSUS is designed such that the currently used WfMS may be substituted for another WfMS. To validate this design, we are currently working on a new version that uses BEA's WLPI system [BEA02].

4.8 Related Work

In this section, we present the two approaches that we think are most closely related to the CONSENSUS approach.

The approach by Su et al. [SHH00] is based on the idea that a consumer registers on a proxy negotiation server by giving a description of the goods or services she wants, her preferences, and a negotiation strategy. Then the server takes over and looks for a supplier that matches the consumer. When it finds one, it starts a bargaining type negotiation until eventually a deal is reached. The server uses the negotiation strategy supplied by the consumer. Notice that in this case, the merchant brokering phase is the responsibility of the server, and that the user has no control over the negotiation. Many negotiations can be started on the same proxy server, and they will be stored as persistent objects in a database.

With this solution, the user will have no control over the negotiations once they are started. She can only see their progress, but cannot intervene in them. We prefer the user to be in control and intervene whenever she wants in the CN. The workflow representing the CN would have to be constructed, stored, and executed at the negotiation server (a feature that is not supported by the proxy server as it is

described in [SHH00]). We prefer the workflow representing the CN to be constructed, stored, and executed on the client side rather than on the negotiation server side. In contrast to our solution, the proxy negotiation server approach does not explicitly support CNs. Furthermore, our solution is decentralized and supports any type of negotiation, whereas the server solution supports only bilateral bargaining.

When we compare the services expected from CONSENSUS to those offered by the KASBAH system [CM96], we see some similarities. The agents can be monitored by their creator via GUI tools. They also can have a negotiation strategy. For KASBAH, the strategy is rather simple and is chosen from a set of predefined strategies. In CONSENSUS, however, the strategy is entered rule by rule. Consequently, it can be more sophisticated and may not be guessed easily by the server or the other participants. In the KASBAH system, for instance, it is possible to guess the negotiation strategy of an agent just by observing it for a while. An agent for which the strategy is known by its opponents is at a disadvantage [BS97]. The agents in KASBAH are mobile, whereas ours are not. The agents in KASBAH conduct only one style of negotiation, which is bargaining, whereas ours will practice any kind of negotiation.

4.9 Conclusion

In this paper, we stated the problem of a CN and showed the need for a CNSS to solve it. A CN involves negotiating many interdependent items by engaging in different and independent negotiations. CONSENSUS is a CNSS, which helps the user model, track and monitor a CN. Its architecture is based on agent technology, as well as workflow management and rule engine technology. CONSENSUS can be used by a consumer from her home computer to negotiate, for instance, a vacation package (B2C), or it can be used in B2B, for example, by a travel agency to negotiate travel packages for its customers.

We claim that a CN can be specified using a workflow. We are certain that a workflow can capture the sequencing of the negotiations and some dependencies between them (we call this the CN know-how), and currently, we are experimenting

with more complex examples to further substantiate our claim. The software agents know the rules of the individual negotiations in which they are involved (we call this individual negotiation know-how). But knowing the rules of the negotiation does not make you a good negotiator. What is needed are negotiation strategies to be used by the agents in order to make them better negotiators (we call this negotiation strategy know-how).

As future work, we aim to further investigate several research issues that are central to our solution. One such issue is how to instantiate software agents based on a description of the negotiation rules. Another issue is the high-level specification language we would like to define and implement in order to free the user from the task of modeling a workflow. The question of breaking commitments in a negotiation will also be investigated along with its effect on CNs. Negotiation strategies are an important research area for us. We will have to come up with strategies that apply to individual negotiations and strategies that apply to a CN. We will also have to see to what extent the negotiation strategies depend on the negotiation type. Our conceptual work will be completed by proof-of-concept prototypes, as incepted with CONSENSUS version 0.1.

In conclusion, we believe that a CNSS such as CONSENSUS is a much-needed tool for coping with the ever-increasing scope and complexity of e-negotiations.

CHAPTER 5

GENERIC NEGOTIATION PLATFORM

Preface

There are two types of negotiation systems, namely negotiation servers and negotiation applications. Negotiation servers (also referred to as negotiation engines) are typically systems that can run multiple negotiations, while negotiation applications usually provide only a single form of negotiation [Wri97]. In this chapter we present GNP (Generic Negotiation Platform), our negotiation server.

Some of the many objectives of the TEM project are to investigate open protocols for electronic marketplaces, explore concepts and tools for e-negotiations, and design new negotiation protocols. At the time we joined the project, there was already a tool called GEE (Generic Experimentation Engine) that supported game-oriented experimentation and allowed for the study of human behavior under various game situations. Yet, there was a need for a complete negotiation server. We also needed a platform to test CONSENSUS, with our two main requirements being (1) support of several negotiation protocols, and (2) an API so that our software agents could connect to it and participate in the negotiations.

GNP was built as a reengineered extension of its ancestor GEE. Typically, a session in GEE (i.e., a game) is made of rounds. Each round consists of receiving the players' responses, deciding upon the outcomes of the round and finally communicating resulting information to the players while preparing for the next round. As an extension of GEE, GNP supports various kinds of economic games. New services are provided to allow for the customization of negotiation rules and error recovery. New persistency management features allow for negotiation execution to be traced and analyzed. A new fine-grained time management to deal with precise and flexible delay evaluation was also provided in GNP.

GNP is designed to run any kind of negotiation. It simply requires the negotiation to be described as a script and a set of data files, which together provide the parameters and the rules of the game. Scripts are written in JPython. In this context, a negotiation designer should ideally have both programming skills and economic market design knowledge [NBB+02]. To facilitate the description of negotiation rules, GNP provides a tool to gather the required information and enter the various negotiation parameters. This tool greatly alleviates the design task by querying the designer for typical negotiation information such as: the attributes and defaults values of the formalized concepts; the condition for ending the rounds, phases and the entire negotiation; and the information to be displayed or hidden from the players. This information is gathered into XML files, and a JPython script is edited to describe the appropriate negotiation rules. GNP uses the JPython script and the XML file to carry out the rounds of the negotiation.

The design of GNP also calls for leveraging the formal descriptions of negotiation rules (see Chapter 3) to provide a repository of negotiation process descriptions, which could be described using UML statecharts. The actual scripts that implement the negotiation (i.e., its economic rules) could be automatically generated from the statecharts, thus making GNP even more generic.

We contributed to the design of GNP by exploring the state of the art in negotiation protocols and negotiation servers, and by participating in defining and refining its requirements. These requirements include an API for software agents to connect to the platform and exchange offers and counter-offers, as well as genericity in a sense that various negotiation types can be implemented with minimum effort using GNP. We also participated in the validation of GNP by using it extensively, by implementing several negotiation protocols and adopting them in the nodes of Combined Negotiations, and by conducting agent tournaments using CONSENSUS.

The remainder of this chapter is a modified version of the following paper:

Towards a Generic E-Negotiation Platform, by Morad Benyoucef, Rudolf K. Keller, Sophie Lamouroux, Jacques Robert, and Vincent Trussart. In *Proceedings of the*

Sixth International Conference on Re-Technologies for Information Systems, pages 95-109, Zurich, Switzerland, February 2000. Austrian Computer Society.

Towards a Generic E-Negotiation Platform

Abstract. To investigate the various research questions concerning e-negotiations, an appropriate software infrastructure is needed. As part of our project on electronic marketplaces, we have developed the Generic Experimentation Engine (GEE). GEE supports game-oriented experimentation and allows for the study of human behavior under various game situations. The more recent Generic Negotiation Platform (GNP) is a more focused version of GEE that will support experimentation with alternative market designs and with various types of negotiations. GNP is being re-engineered from GEE in that important concepts and technologies developed for GEE are carried over into GNP. The paper provides a detailed overview of GEE. The re-engineering of GEE into GNP is addressed by presenting a list of lessons learned from the GEE development and by providing a vision for GNP. Furthermore, the paper addresses the formalization of auction rules, one of the prerequisites for making GNP truly generic.

5.1 Introduction

The rapid and significant progress of information technologies is profoundly changing the structure of the economy, in particular the way economic negotiations are undertaken and exchanges of goods and services are organized. Electronic marketplaces as nexus of services for the largest possible network of businesses will be at the core of future e-commerce. Negotiations will be supported by open negotiation servers where deals are struck and prices determined, and where evaluation, matching, and advising services are being offered. The creation and efficient operation of electronic marketplaces raises many challenges. E-commerce should do more than just replicate what is being done today, and gains may only be obtained through re-engineering the way decisions are made and markets operate. In the TEM (Towards Electronic Marketplaces) project, a joint industry-university project started in early 1999, we aim to make the above vision possible. Thus, we

address market design issues in respect to resource allocation and control and reward mechanisms, investigate open protocols for electronic marketplaces, and explore concepts and tools for e-negotiations. E-negotiations may be re-engineered from traditional negotiations, or designed expressly for electronic marketplaces.

As part of the initial phase of the TEM project, we have built GEE (Generic Experimentation Engine), a software prototype for game-oriented experimentation. In its current version, GEE supports continuous double auctions [WWW98b], and is fit for the study of human behavior under various game situations. In the recently incepted phase II of the project, we are developing GNP (Generic Negotiation Platform). Whereas GEE is a game-oriented engine, GNP is an auction-oriented platform, that is, GNP will be larger in scope and will support many different types of negotiations (auctions are considered a special case of negotiations). The different types of negotiations are described in a uniform and formal way, and the descriptions can be serialized for exchange in the e-marketplace. From an engineering point of view, GEE is being re-engineered into GNP, which will exhibit a layered architecture based on objects, statechart diagrams [UML98], and scripts. Our development process is incremental and follows the feature development approach described in [CLD99]. It is the re-engineering at the infrastructure level, which is the subject of this paper.

The contributions of this paper are threefold. First, the paper provides a detailed overview of GEE. Second, the re-engineering of GEE into GNP is addressed. This has been done by factoring out and parameterizing the application-independent components of GEE so that new applications can be developed easily and quickly. Our approach is similar to that taken by expert system shells and by high-level programming environments such as 4GL and SQL, yet it has not, to our knowledge, been applied to the e-negotiation domain. Third, the formalization of auction rules is discussed, and the approach adopted in TEM is explained.

In Section 5.2 of this paper, the requirements for TEM's experimentation engine and negotiation platform are discussed. Section 5.3 gives an overview of the

implementation of GEE. In Section 5.4, the lessons learned from the GEE development are explained. Then, in Section 5.5, the need for formalizing auction rules is discussed, and the TEM approach is presented. Section 5.6 provides our vision of the GNP functionality and architecture. Finally, we wrap up the paper with some concluding remarks.

5.2 Requirements for Experimentation Engine and Negotiation Platform

The level of abstraction required by a generic e-negotiation platform is quite high. The negotiation engine should be able to support a wide variety of disparate negotiation rules and algorithms. The common ground shared by these algorithms is that they can be viewed as an economic game according to game theory.

Game theory, an important branch of microeconomics, provides a mathematical structure to study social interactions among rational agents (forward-looking, Bayesian and optimizing). An extensive-form game is an exhaustive description of the social interaction under consideration. In order to be well defined, a game must specify:

1. the set of players;
2. a sequence of decisions;
3. the precise structure of the information flow;
4. a representation of the players' preferences over the set of all possible outcomes of the game.

Hence, it must specify what each player knows at the beginning of the game, in particular what information is public and what is private; by whom and when each decision is made and what information the decision-makers have when making their decisions; and finally, what are the gains (monetary or otherwise) accruing to the players as a result of the history of the game.

The precise definition of the extensive representation of a game can be found in graduate micro-economic textbooks (see, for instance, [MWG95]-Chapter 7, or

[KW82]-Section 5.2). In order to implement a game, we can proceed through a finite number of rounds. In each round, one or many participants are provided with some information (in the form of a table or a small text) and are invited to select an action. This generic structure is sufficiently general to implement any finite game. We can implement simultaneous-move games in which players are asked to select their actions simultaneously, or sequential games in which players are informed of past plays and are asked to choose moves one at the time. The key is that we are able to control both the information that participants receive and the set of actions, which they are allowed to select.

Our objective was to build a generic experimentation engine that is flexible, so it can implement any conceivable game. In order to do so, GEE's unique responsibility is to orchestrate a sequence of rounds. In each round, including the initial one, each player receives sets of public and private information and of selectable actions. Depending on the specific design of the game, a new round is initiated either when a predefined number of players have submitted their choices or when a specific delay has expired. At the beginning of every round, the sets of public and private information and of selectable actions are updated according to the rules of the game. Finally, at the end of the game, each player receives a score that can eventually be translated into a monetary gain. To help players keep track of how well they are doing, points are added to or subtracted from their score after each round. The flexibility of GEE comes from the fact that it imposes no specific restrictions. How the private and public information, the selectable actions and the gains are initially generated and updated after each round is not part of the GEE design. It is handled as configuration information for the specific instance of the experimentation engine.

Market negotiation games form a special class of economic games. The major characteristic of market negotiation games is that their outcome mainly specifies two things: (1) a final allocation, which determines who gets the item or items on trade; and (2) prices or monetary transfers specifying who pays or receives what. The negotiation process includes, depending on the specific design, rounds of offers and

counter offers (quotes, bids, asks, etc.) and a closure rule, which ultimately leads to an "allocation and prices" outcome. GEE was developed in order to have an open electronic environment that could reproduce any mechanism involving several players interacting with each other. It was done with two objectives in mind: (1) to be a functional and flexible tool to validate economic theory by making electronic experimentations on small groups of individuals; and (2) as we focus on negotiations, to include some kinds of negotiation rules in GEE to help extract the common concepts found in negotiation processes.

The objective of GNP is to offer a more focused version of GEE that will enable us to experiment with various market designs. Using GNP, it should be easy to create and deploy various types of negotiations. For such easy deployment to be possible, as is already the case with GEE, no client software installation should be required (except for a web browser).

The development of GNP as a platform supporting multiple negotiation algorithms will lead us to formalize and validate common concepts and similarities and to develop a clear and consistent terminology for negotiations. Such knowledge will help us create an open protocol for negotiation information exchange and a formalized representation of negotiation rules. Eventually, GNP should be powerful and flexible enough to allow further research and development in a wide range of fields, including operations research and decision support systems.

To reach these goals, we adopted an iterative approach based on feature development [CLD99]. The choice of this software engineering process is guided by the fact that our requirements are evolving continuously. Our first prototype, GEE, is an implementation focusing on the creation of games according to game theory principles and on flexibility. GNP is meant to be larger in scope and to address the specifics of negotiations. Finally, concepts and base technologies developed for GEE should be reusable when building GNP.

5.3 Implementation of GEE

In this section, we present our first prototype of GEE. The section is organized into the subsections user inter-face, dynamic behavior of games, data management and scripting, and architecture and technology.

5.3.1 User Interface

GEE is an application accessible via the Internet. If GEE is activated and some games are already on, a player newly logged on can choose and participate in these games. A game process is divided into rounds. Arriving at any time, a player will get an HTML page that will not change during the current round. The page is composed of four sections (see Figure 5.1, left). Section 1 contains information about the current round (time, round number, etc.). Section 2 shows information in read-only mode. This can be public information about the state of the game given to all players and/or private information given only to individual players. The information in Section 2 is an HTML fragment generated by a script and inlined at the beginning of a round. Section 3 displays questions in the form of a questionnaire. By completing and sending out the questionnaire, a player participates in the current round. The responses of the players will only be treated at the end of the current round, and feedback will be given at the beginning of the next round, by displaying updated information and new questions in the HTML page. A game is over when no more questions are generated and instead a "final message" is issued to all players. Section 4, finally, is an icon bar that provides access to other useful functions, such as help files and transactions history.

Figure 5.1, right, shows a screenshot taken during one of the rounds of a game. The game in this example is a continuous double auction. The player can submit a *bid* (buy order) or an *ask* (sell order). Section 3 lets the player choose to add or delete an order, and to buy or sell, and enter the quantity and the price. Section 2 displays the orders submitted by all the players (orders submitted by the player interacting through the HTML page are tagged with an asterisk *) and the current price (last trade). Furthermore, Section 2 shows the wallet of the player, that is, the number of units and

the amount of money of the player. Finally, it also depicts the wallet when the engagements made by the player are taken into consideration, and it displays the details of the last transaction. Sections 1 and 4 are not shown in the figure.

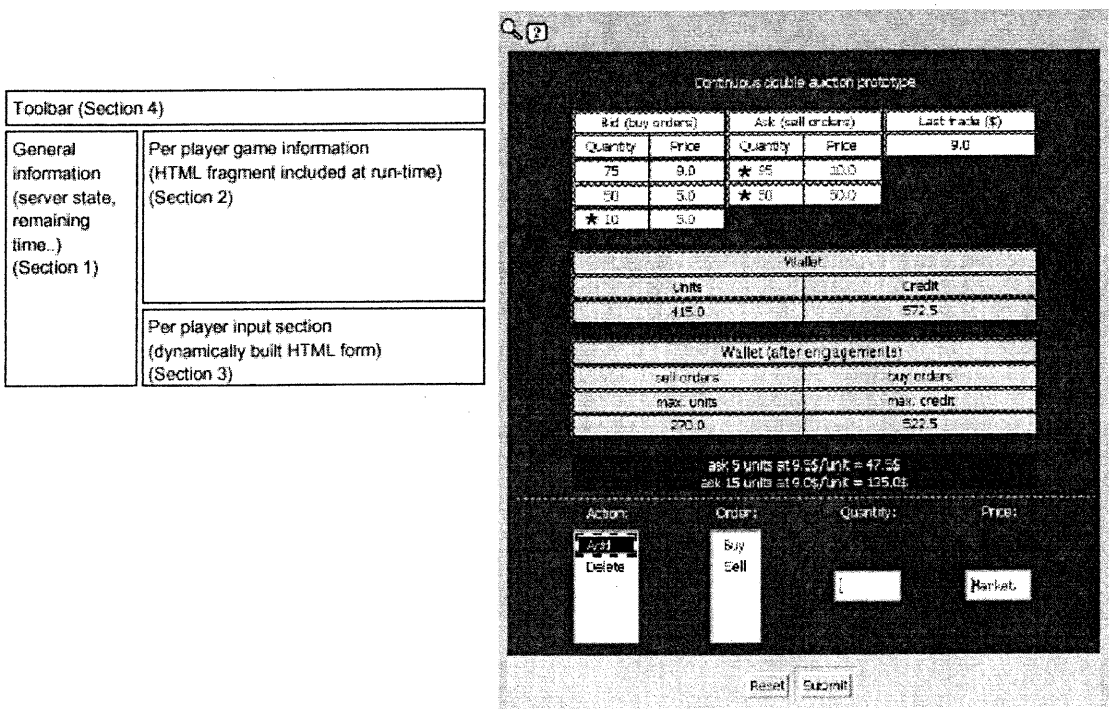

Figure 5.1: User interface of GEE: schema (left) and screenshot (right)

5.3.2 Dynamic Behavior of Games

The statechart diagram shown in *Figure 5.2* captures the behavior of the games supported by GEE. Games are based on an iterative pattern that we call a round. During a round, the responses sent by the different players are queued. A round ends on a particular event, which can be either the number of responses sent, a timeout, or both. When such an event occurs, a file containing a script program is called which processes the queued responses and generates the messages (information, questions) sent to the players for starting the next round.

Rounds, information, questions, and responses are the basic concepts built into GEE. The game designer can use these concepts in a flexible way, having full control over

the way data is computed from one round to the other. We define a game designer as the person who sets up the rules of the game and implements them in a script.

5.3.3 Data Management and Scripting

The data of GEE are managed in files and directories. Specifically, GEE is in charge of the output files (information and questions) to be displayed on the player's screen and of the input files (responses) to be written in the directory assigned to the current round. Once an end condition is met, an executable file, that is, a script file written in JPython [Jpy02], is called to complete the current round. The script processes the input files produced during the ending round and generates output files into the directory assigned to the next round.

A round's end condition may vary from round to round. End conditions are stored in dedicated files that can be manipulated by the scripts. The activity diagram [UML98] shown in *Figure 5.3* summarizes the life cycle of a script for a given round.

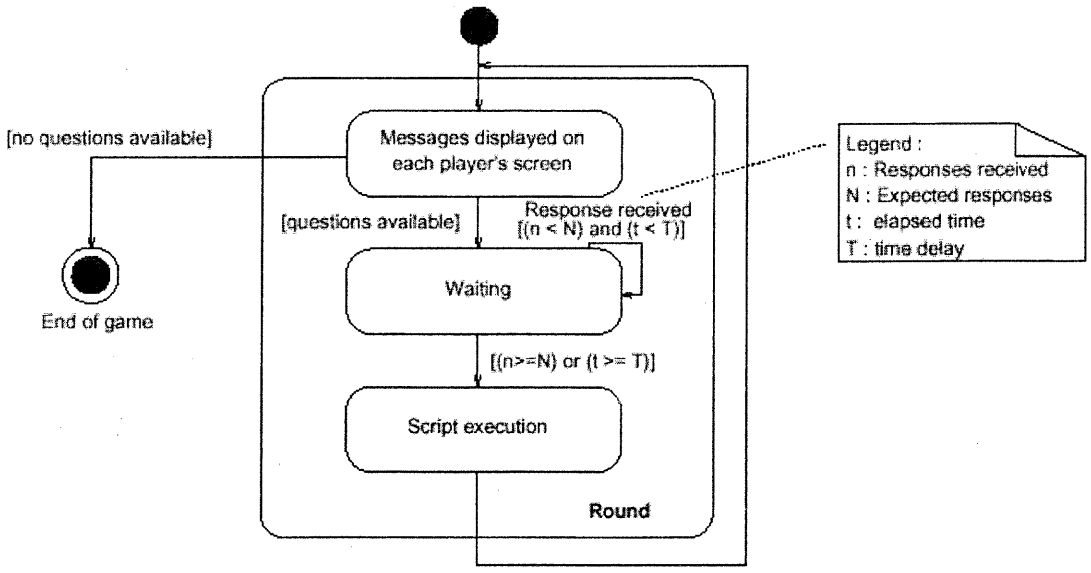

Figure 5.2: Statechart diagram of games supported by GEE

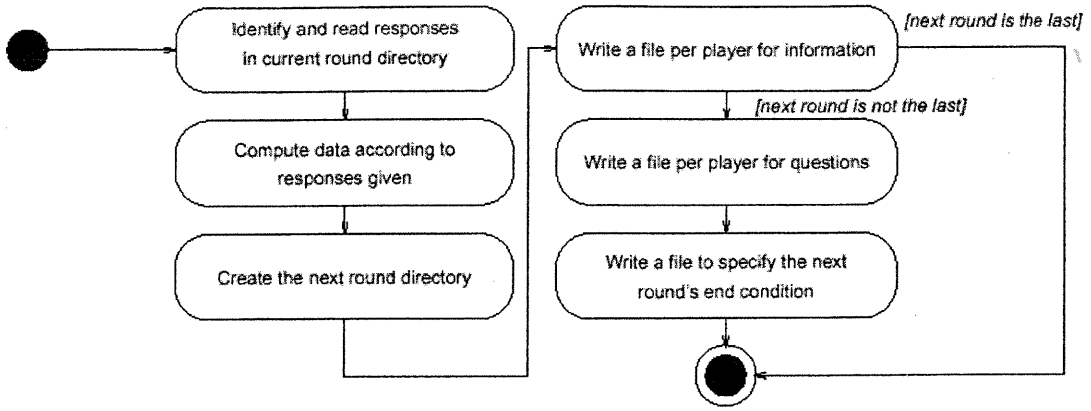

Figure 5.3: Activity diagram of GEE script for a given round

As building a directory hierarchy and scripting are under the control of the game designer, anyone with knowledge of JPython should be able to set up a game with her own rules. During scripting, the game designer has to decide on which round will be the last. In the second but last round, the script should not write any questions to the players, nor specify a round's end condition. The absence of questions will be interpreted by GEE as the end of the game.

5.3.4 Architecture and Technology

To ensure easy deployment, the negotiation engine is built as a web application using server-side Java technologies that is accessible in three ways (see Figure 5.4):

- Pull mode: the information is sent to the player's web browser only upon request. In order to keep the displayed information up-to-date, this mode relies on polling a dynamically generated web page. The pull mode is lightweight; however, it is little responsive and imposes the task of browser reloads onto the player.
- Push mode: in this mode, the information is sent from the server to the client when needed. This allows for a responsive and full-featured user interface at the expense of a heavy download (Java applet) and uncertain reliability (Java 1.1 is currently not well supported in the major web browsers).
- Hybrid mode: this mode sits in between the two previous ones. It relies on a very small Java applet to perform on demand web page polling by listening for events

on the server (through a TCP/IP socket) and performing a browser reload only when needed.

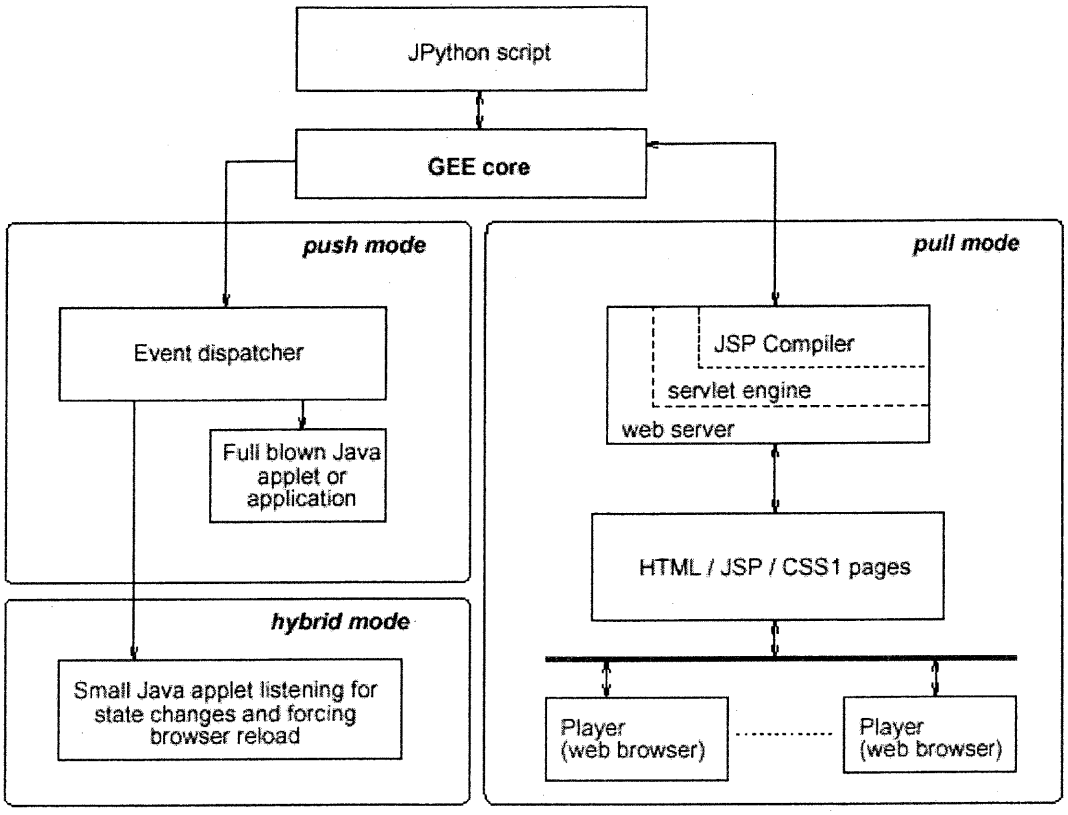

Figure 5.4: GEE architecture

The selection of the access mode is a task performed at deployment time by the operator of GEE. The preferred access mode is the hybrid mode because of its lightness, responsiveness, easy development and customizability.

The web pages for the players are generated dynamically (cf. Figure 5.1). Since these pages are coded using the Java Server Pages (JSP) specification [JSP00], multiple deployment alternatives are provided.

5.4 Lessons Learned from GEE Development

The GEE prototype is currently being used for experimentation of economic games. Our development experience and the ongoing field tests have shed light onto the strengths and weaknesses of the design and implementation of GEE.

In this section, we present some of the shortcomings of GEE and how we intend to overcome them in GNP. We introduce the platform services that will be provided, a new approach to time management, a negotiation toolkit that will be made available, and a high level interface for easing the task of the game designers.

5.4.1 Platform Services

GEE only provides basic error handling. In GNP, we envision pervasive error handling in respect to the players, the game designer, and the person operating GNP. Furthermore, a registration tool to regulate and manage access to GNP and to customize negotiation rules according to the profiles of players will be provided. Finally, persistence management in GEE is done manually by the game designer. In GNP it will be a built-in feature supported by appropriate technology. This will allow recovery after game failures and for tracing for recording and analysis purposes.

5.4.2 Fine-grained Time Management

Time in GEE is managed merely at the round level, limiting every round with a time delay. As time is a decisive factor in negotiation processes, it should be dealt with in a more flexible way. First, a delay should limit the overall duration of a negotiation. Second, this overall delay should be divided into several limited time periods. Those periods would represent different phases of a negotiation process, e.g., a number of consecutive rounds, in which the same rules could be applied. Such fine-grained time management would allow for the more precise definition of negotiation processes. Moreover, relative time delays as well as absolute dates should be supported.

5.4.3 Negotiation Toolkit

The architecture of GEE is independent of any specific negotiation type, with data files and scripts being the only entities describing the rules of the games. In complex games and negotiation processes, the size of the files and scripts may become rather large. Especially, scripts may become difficult to write and maintain. Since the scripts are written in JPython, we were naturally inclined to build many generic reusable

classes, persistent objects, and files for use in different negotiation processes. A sample of these is:

- initial data of a negotiation: announce;
- information sent to all players: market quote;
- information sent to some players: private message;
- choices that the players make in each round: bid;
- matching of bids and asks according to negotiation rules: transaction.

Whereas in GEE this collection of classes was built in an ad hoc manner, GNP will provide them from the outset organized as a toolkit.

5.4.4 High-level Interface for Game Designers

In GEE, a game designer who wants to implement a game has to manage all the data and is responsible for writing the scripts. In the case of negotiations, she would need substantial knowledge of both programming and economic market design. Even if a powerful, simple and easy to learn language such as JPython is used, programming may become a burden. A good knowledge of market design should be the only prerequisite for effectively using GNP.

We have identified a number of operations that are common to different negotiation processes. Some of these operations are:

- define attributes and default values for the formalized concepts;
- setup the end conditions for rounds, phases and the whole negotiation;
- define the information to be displayed to or hidden from the players.

Such operations may be provided to the game designer as a high-level interface. In the medium-run, however, we want to free the game designer from directly dealing with this high-level interface and any negotiation toolkit. To this end, we envision to leverage the formal descriptions of auction rules, as described in the section below.

5.5 Description of Auction Rules

5.5.1 Need for Formalization

If we are to make buying and selling decisions based on automated auctions, then it is important that we have high confidence in the software involved in this activity. Cass [CLL+99] suggests that an automated negotiation must have four necessary properties. It must be:

- Correct: an automated negotiation must not contain errors such as deadlocks or incorrect handling of exceptions;
- Reasonable: it must allow adequate time for bids to be made (or for decisions to be made in general);
- Robust: it should continue to function properly after an improper action by the user;
- Fast: it has to execute and respond quickly.

We believe that there should be a fifth property. An automated negotiation must also be traceable. In order to be trusted, the software must be able to justify its actions to the user if necessary. This could be done by tracing the execution and by showing the decisions taken together with their rationale [GMM98].

In order to achieve these properties, we need to formally describe negotiation processes. To our knowledge, this issue has not been explored yet in any depth. In the literature, we found few papers addressing such formalization. Typically, natural language is used to describe the negotiation and auction types under consideration. Some papers try to classify them while others try to find similarities so that common parts can be isolated [BK99]. We agree with Cass [CLL+99] that "a notation with well-defined, well-formed semantics can facilitate verification of desirable properties".

Another issue is the need to separate the process of negotiation from the other parts of the software. We believe that the rules governing the negotiation should not be

hardcoded. The software tool supporting the negotiation should be able to pick one type (style) of negotiation from a repository and use it. Separation permits efficient implementation, easy testing and, last but not least, encourages reuse. We can actually benefit from the fact that negotiation processes contain parts that are common to all of them.

Another reason we need a formal way to describe the auctions is that the auction rules should be known to the user (human or software). They should be available to anyone who wants to consult them before engaging in the auction. We think that the rules are as important as, or perhaps even more, than the other information describing the good or service that is the object of the auction. We agree with Wurman [WWW98a] that "implementing auction mechanisms in a larger context will require tools to aid agents in finding appropriate auctions, inform them about auction rules, and perform many other market facilitation functions".

5.5.2 Defining a Formalism for Auction Rules

We need a mechanism that enables us to formally describe the rules governing an auction, visualize this description when necessary and serialize it in order to transfer it over the network. A review of the formal methods used to describe auctions can be found in [BK99].

Wurman et al. [WWW98a, WWW98b, WW98, Auc00] categorize auctions as either single or double, sealed-bid or open-cry, and ascending or descending. They use messages and activities to describe an auction. The three main activities are: receive-bid, clear, and supply-information. Parameters are used to complete this description. These parameters indicate for example if a new bid should be greater than the previous one, what are the closing conditions, etc. Natural language is used to describe the auctions.

Kumar and Feldman [KF98a, KF98b] use a Finite State Machine (FSM) to model an auction. The states of the FSM are the states of the auction. Its input alphabet is the set of messages that can be sent by the participants. A message is expressed as a pair

< p , m > where p is the sender of the message and m is the message sent out. The output alphabet is the set of messages sent to the participants. These messages are expressed as pairs <<p , m >> where p is the subset of all the participants that will receive the message and m is the message itself. The process flow of the auction maps into the transitions of the FSM. The messages make the auction go from one state to another.

The FSM alone is not sufficient to describe an auction process. Therefore, the description is fine-tuned according to the following policy decisions: anonymity, restriction, rules for closing the auction, and services provided to the participants.

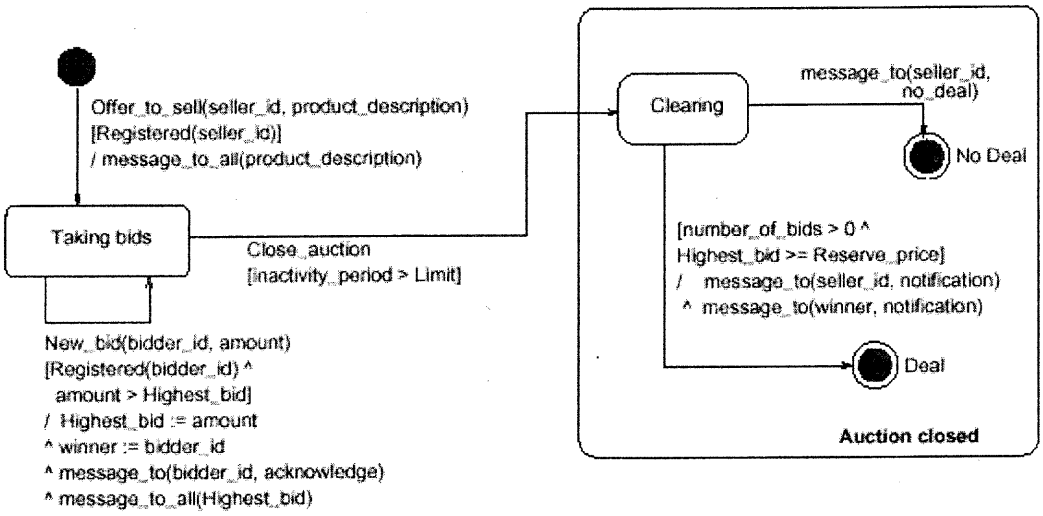

Figure 5.5: Statechart diagram of an English auction

Little-JIL [Wis98] is a graphical agent coordination language realized at the University of Massachusetts at Amherst. A program in Little-JIL has a root that represents the entire process. The root itself is decomposed into steps that describe the process. Visually a step is a rectangle with many tags associated to it (the name of the step, the resources necessary to execute it, etc.). The execution of a step and its sub-steps is controlled by the step's sequencing (sequential, parallel, conditional, etc). Little-JIL was used to describe a set of auction processes [CLL+99]. This way of visualizing an auction process simplifies reasoning by following the flow of control

and the flow of data in the tree of steps. Since an auction involves multiple participants, Little-JIL may be used to decompose the auction into steps to be carried out by the participants and by providing coordination and communication between them. This notation is unfortunately quite exotic, and the programs are hard to understand.

The only formalism that satisfies most of our requirements is Kumar's and Feldman's FSM. However, the FSM alone cannot capture the complete auction process. We propose to extend it by using UML's statechart diagrams. They are well established and widely used, and they are semantically rich enough to formally describe and visualize processes. An important feature statechart diagrams have and that we will be relying on is the possibility to be serialized in XMI [XMI98]. Various types of auctions can indeed be described using statecharts, including the Continuous Double Auction (CDA), which is implemented in GEE, and the English auction which is described in Figure 5.5. The main states of the English auction are "Taking bids" and "Auction closed". When the auction is closed it goes to the "Clearing" state where the auctioneer has to determine if there is a deal or not. The two possible final states are "Deal" and "No Deal". In the first case the seller and the buyer are notified. In the second case only the seller is notified. The transitions are labeled with the string event[guard-condition(s)]action(s).

5.6 GNP Vision

GNP builds upon the lessons learned from GEE and shifts focus from games towards electronic negotiations. As GNP will be designed especially for negotiations, implementing a negotiation through scripting will be even simpler than creating a game with GEE.

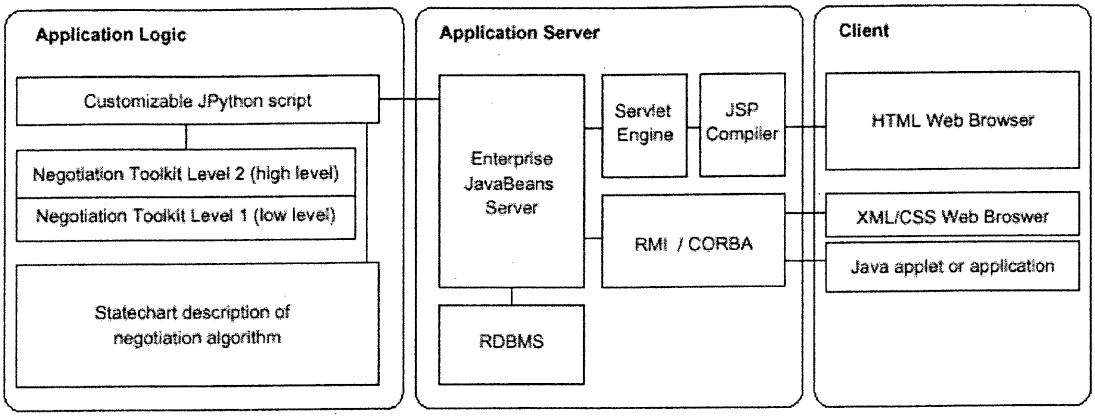

Figure 5.6: GNP architecture

Two new services will be provided to the game designer: a negotiation rule description analyzer that can process a formalized description of negotiation processes and a toolkit (a collection of classes) implementing frequently needed concepts (wallet, information storage, etc.). For example, a game designer could create (or reuse) a formalized description of a Dutch auction using a statechart diagram serialized using the XMI format and script a negotiation using this description (see Figure 5.6).

To further improve the openness of the negotiation platform, an information exchange protocol based on XML [XML98] will be developed. Such a protocol will allow for the creation of software agents that can be used as smart advisors, and for setting up a simulation environment to investigate optimal strategies for complex negotiation algorithms.

The architecture of GNP will reuse several concepts and technologies that were already adopted in GEE: it will be a multi-tier web application using servlets, JSP pages and JPython scripts. However, GNP will need to support large scale deployments and features required by an enterprise level electronic negotiation platform, such as security, scalability, reliability, persistence, and transactional behavior. To meet these requirements, GNP will be a reusable server component executing in an Enterprise JavaBeans (EJB) [EJB00] application server using a Relational Data Base Management System (RDBMS) for information storage.

GNP will be document oriented, that is, a negotiation is treated as a document, and every interaction is a manipulation of that document. This approach eases the scripting task since the internals of the system are open and accessible. To help manipulate these internal documents, a two level API will be provided which we call the Negotiation Toolkit. The first part of the toolkit (referred to as NTK-1) consists of storage management functions, such as loading, searching, and saving information about the negotiation. The second part (NTK-2) will provide functions and objects that are frequently needed by the game designer, according to our experience with GEE. For example, the following objects will be provided by NTK-2: Bid, Negotiation, Round, Player, Wallet, etc. Another benefit of the document-oriented approach is that obtaining an XML formatted document containing information about a negotiation is trivial; this will allow us to easily create an open protocol based on XML.

Since the most recent generation of web browsers can process and format XML documents combined with CSS style sheets, the XML protocol for negotiation information exchange will lead to more responsive user interfaces, without affecting their low cost of development and deployment. For example, a small Java applet loaded in the player's browser could listen for events on the EJB server (using RMI or CORBA [Sur98]) and update only portions of the DOM tree [DOM00] contained in the browser without requiring a complete web page reload.

5.7 Conclusion

In this paper, we detailed GEE, a game-oriented experimentation engine, and described how GEE will be evolved and adapted to become GNP, a generic negotiation platform. We have shown that in this endeavor several important concepts and technologies of GEE will be reused, allowing us to come up with a GNP prototype within just a few months.

There is already some evidence from experimentation with 10 students that indeed GEE constitutes a flexible, powerful, and useful infrastructure for studying the

behavior of players under game conditions. After several iterations, the design of GNP is now stable enough to start implementation.

Once GNP is available, various research questions can be investigated. A sample problem from operations research that will be addressed using GNP is determining the winner in a combinatory auction. In combinatory auctions, the user may bid on a combination of items, and the problem for the auctioneer is to determine a winner, that is, the bid that maximizes the auctioneer's gains. Another class of negotiations, combined negotiations, will be refined using GNP. Combined negotiations allow an agent (human or software) to engage in and manage many negotiations at the same time. This latter type of negotiation will lead us to explore the role of decision support systems and workflow management systems [LW99] at the infrastructure level. For instance, in the case of business-to-business e-commerce, such systems may carry out after-the-deal activities.

The wide range of the above research questions underlines the value of providing GNP. The dynamic nature of this research suggests a feature development approach, and an ongoing re-engineering of GNP. Given the positive experience of evolving GEE and designing the first GNP prototype, we are confident that GNP will prove flexible and useful throughout the project.

CHAPTER 6

RULE-DRIVEN NEGOTIATING SOFTWARE AGENTS

Preface

In this chapter, we detail an important facet of CONSENSUS, which was introduced in Chapter 4, but without much detail, namely the representation and management of the knowledge that defines the behavior of the software agents.

CONSENSUS is built around a Workflow Management System (WfMS) that is responsible for dispatching work to the agents according to the sequencing information supplied by the user. Once the agents are started, (i.e., they are given control by the workflow engine), how would they behave? We divided the behavior of the agents into three components: protocols, strategies, and coordination.

The protocols are the negotiation rules that every agent should follow when interacting with the negotiation server (i.e., when to submit an offer, how and when to respond to a counter-offer, when the negotiation ends, etc.). CONSENSUS instantiates the agents according to the protocol supported by the negotiation server (English, Dutch, etc.). We showed in Chapter 5 how GNP implements various negotiation protocols.

An agent's negotiation strategy is the specification of the sequence of actions (usually offers and responses) that the agent plans to make during the negotiation [LMJ00]. An agent's coordination, on the other hand, is the knowledge required in order to coordinate its actions with those of the other agents participating in the Combined Negotiation.

An important part of our research was to come up with a mechanism to represent negotiation strategies and coordination. Being aware of the fact that most existing agent-mediated e-negotiation systems rely on hardcoded schemes to represent

strategies, we chose a rule-driven approach to represent, manage, and explore strategies. The advantages of this choice are the high level of abstraction of the rules, their closeness to human understanding, their versatility, and their ability to be modified at run time. We also chose to represent coordination knowledge as if-then-rules for the same reasons we stated above. In addition to these advantages, our representation was also in response to the first two architectural objectives set out in Chapter 1 (Section 1.3, (5)a. and (5)b.). The solution is indeed realistic, and is implemented using off-the-shelf rule engines.

Plenty of time was spent on validating this architectural choice. To this end, several negotiation protocols were implemented on GNP, with a focus on the English auction, the Dutch auction, and the multi-item Dutch auction.

We devised bidding strategies, coded them as rules, supplied them to the software agents, used CONSENSUS to conduct agent tournaments, and watched and documented their behavior. Similarly, we devised and tested coordination schemes between agents.

Using this approach, we demonstrated that we could capture a wide range of bidding strategies and bid coordination schemes, using a rule-based approach, while supporting a wide spectrum of negotiation types. This was tested in agent tournaments within simulated markets, and the results are encouraging.

This work led to two publications [BAK01] and [BAL+02]. Hakim Alj participated in the implementation of the system as part of his Master's thesis, and Kim Levy helped with the agent tournaments.

In the remainder of this chapter, we present a modified version of the following paper:

A Rule-driven Approach for Defining the Behavior of Negotiating Software Agents, by Morad Benyoucef, Hakim Alj, Kim Levy, and Rudolf K. Keller. *In Proceedings of the Fourth International Conference on Distributed Communities on the Web,* Sydney, Australia, April 2002. Springer. LNCS. To appear.

A Rule-driven Approach for Defining the Behavior of Negotiating Software Agents

Abstract. One problem with existing agent-mediated negotiation systems is that they rely on ad hoc, static, non-adaptive, and hardcoded schemes to represent the behavior of agents. This limitation is probably due to the complexity of the negotiation task itself. Indeed, while negotiating, software (human) agents face tough decisions. These decisions are based not only on the information made available by the negotiation server, but on the behavior of the other participants in the negotiation process as well. The information and the behavior in question are constantly changing and highly uncertain. In this paper, we propose a rule-driven approach to represent, manage and explore negotiation strategies and coordination information. Among the many advantages of this solution, we can cite the high level of abstraction, the closeness to human understanding, the versatility, and the possibility to modify the agents' behavior during the negotiation. To validate our approach, we ran several agent tournaments, and used a rule-driven mechanism to implement bidding strategies that are common in the English and Dutch auctions. We also implemented simple coordination schemes across several auctions. The ongoing validation work is detailed and discussed in the second part of the paper.

6.1 Introduction

According to Kephart et al., over the course of the next decade, the global economy and the Internet will merge into an information economy bustling with billions of autonomous software agents that exchange information goods and services with humans and other agents [KHG00]. In addition to exchanging information, software agents can be programmed to search, compare, learn, negotiate, and collaborate [Jon99], making them particularly useful for the information-rich and process-rich environment of electronic commerce (e-commerce) [MGM98]. As e-commerce usually involves information filtering and retrieval, personalized evaluation, complex

coordination, and time-based interaction (among other things), it can greatly benefit from the introduction of software agents. Therefore, we talk of agent-mediated e-commerce. A simple e-commerce transaction can be seen as a three-phase scenario: (1) finding a partner for the transaction; (2) negotiating the terms of the transaction using a recursive process; (3) and carrying out the transaction. We are interested in the negotiation phase, and particularly in its automation by way of software agents capable of mimicking some of the behavior of human negotiators. We believe that agent-mediated negotiation absorbs many of the costs and inconveniences of manual negotiation [PUF99].

The negotiation process itself is a form of interaction made of *protocols* and *strategies*. The protocols comprise the rules (i.e., the valid actions) of the game, and, for a given protocol, a participant (human or software) uses a strategy (i.e., a plan of action) to maximize her utility [GMM98]. Based on this, many strategy-enabled agent-mediated negotiation systems have been described in the literature. Unfortunately, most of them use hardcoded, predefined, and non-adaptive negotiation strategies, which is evidently insufficient in regard to the ambitions and growing importance of automated negotiations research. The well-known KASBAH agent marketplace [CM96] is a good example of such systems. To overcome this shortcoming, we believe that negotiation strategies should be treated as declarative knowledge, and could, for instance, be represented as if-then rules, and exploited using inference engines.

The focus of our research is on combined negotiations [BAV+01], a case where the consumer combines negotiations for different complementary items that are not negotiated on the same server. For instance, a consumer may want to simultaneously purchase an item and its delivery by engaging in separate negotiations. If software agents are assigned to these negotiations, this poses a coordination problem between them. Many multi-agent negotiation systems found in the literature still rely on ad hoc schemes to solve this problem [IFS+00, Pri00]. Again, we believe that a

declarative approach can be used to describe and manage agent coordination across several negotiations.

To validate our approach, we designed and implemented an automated negotiation system called CONSENSUS [BAV+01] that enables a human user to instantiate one or more software agents, provide them with negotiation strategies and coordination know-how, register them on corresponding negotiation servers, and launch them. The agents use the strategies to negotiate according to the protocol dictated by the server, and the coordination know-how to coordinate their actions. An example of a strategy, applicable to an English auction (one of many existing negotiation protocols) is: "*if you notice any form of jump bidding, then quit*". *Jump bidding* means making a bid that is far greater than necessary in order to signal one's interest in the auctioned item (see Section 6.4). An example of coordination know-how, applicable to two agents bidding as partners in two separate auctions for two complementary items is: "*if your partner looses in its auction, then stop bidding and wait for further instructions*" (see Section 6.4). We are currently testing various strategies and coordination schemes by way of agent tournaments. A large part of the paper is dedicated to this ongoing work.

Section 6.2 of the paper gives some background on strategy-enabled agent-mediated e-negotiations, and then presents our approach. Section 6.3 details our view of agent coordination after reviewing some related work. Section 6.4 is dedicated to the validation tests we are currently conducting. We wrap up the paper with a conclusion in Section 6.5.

6.2 Negotiation Strategies

6.2.1 Agent-mediated E-negotiation

Electronic negotiation (e-negotiation) takes place when the negotiating function is performed by (networked) computers. Fully automated e-negotiation requires that all parties involved be software agents, semi-automated e-negotiation involves a human negotiating with a software agent, and manual e-negotiation refers to processes in which all parties are human [BSS96]. Within fully automated e-negotiation, Parkes et

al. identify autonomous and semi-autonomous agents. The former require complete preferences in order to represent the user, the latter only bid when they have enough knowledge, and query the user when their best action is ill-defined given the current information [PUF99]. Agents can be involved in competitive negotiations when they have conflicting interests. They can be involved in cooperative negotiations when they all aim at satisfying the same interest [BS97].

In addition to their use in e-negotiations, agents are actually used in other phases of e-commerce transactions, including shopping, advertising, delivery, marketing and sales analysis [KHG00]. Pricebots, for instance, are software agents that employ price-setting algorithms in an attempt to maximize profits, thus helping sellers to increase flexibility in their pricing strategies. An example taken from [GK99] points to *books.com*, which implements real-time dynamic pricing by using pricebots to monitor prices on competitor sites and offer the customer a lower price. Shopbots, to cite another example, are agents that gather information from multiple on-line vendors about the price and quality of goods and services [GK99]. One such system is *mysimon.com*.

6.2.2 Challenges of Strategy-enabled Negotiation Systems

Designing, building, and tuning software agents before letting them loose in widely competitive scenarios like e-negotiations, inhabited by (human and software) expert negotiators, happens to be an arduous task [GGR98]. This is why most strategy-enabled agent-based systems use predefined and non-adaptive negotiation strategies in the generation of offers and counteroffers [WZK00]. Some commercial online auction sites such as *eBay.com*, offer the possibility of *proxy bidding* – i.e., an agent with a simple strategy: "*bid until you reach your reserve price, by going up each time with a certain increment.*" On the academic front, the buying (selling) agents of the KASBAH marketplace can choose between three negotiation strategies: *anxious*, *cool-headed* and *frugal*, corresponding to *linear*, *quadratic*, and *exponential* functions, respectively, for increasing (or decreasing) their bid (or ask price) over time [CM96].

Negotiating agents face tough decisions such as whether or not to accept a bid, whether or not to bid, how much to bid, etc. Those decisions must profit from all the information available in the marketplace: description of the good, its expected resale value, price history, participants' identities and behavior, etc. [GGR98]. This information is constantly changing and highly uncertain – new goods become available, buyers come and leave, prices change, etc. To complicate things further, the participants' goals, beliefs, intentions are expected to change over time [NLJ96].

A successful strategy must take into account the strategies of the opponents [GGR98] as well as their reputation [KHG00]. It must also protect against the opponents trying to extract the agent's private information. In a bargaining situation for instance, the buyer agent usually knows its owner's willingness-to-pay for an item [Var95]. If the seller agent (human or software) discovers this information, it can make a take-it-or-leave-it offer that will extract the buyer's entire surplus. Finally, we should mention that, with *eBay.com*'s proxy bidding, one must reveal the highest price one is willing to pay, thus giving the site information that could be used to cheat the bidder [Pri00].

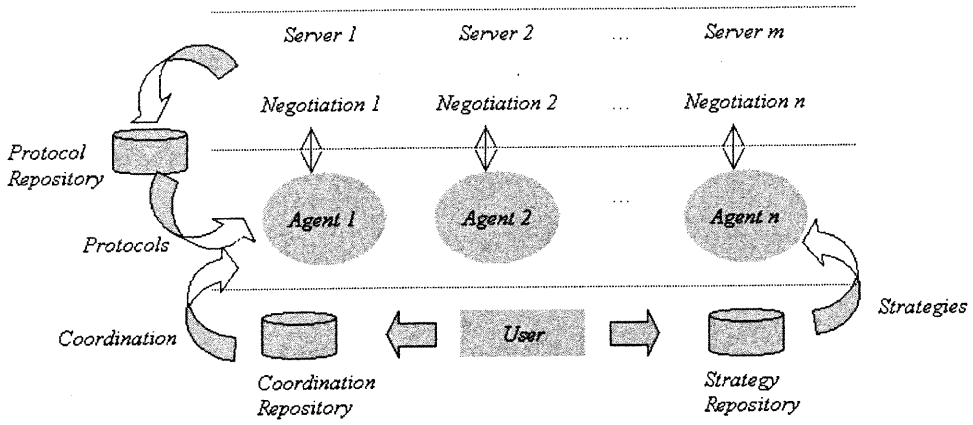

Figure 6.1: Protocols, Strategies and Coordination in CONSENSUS

6.2.3 The CONSENSUS Approach

As mentioned in Section 6.1, the basic components of automated negotiation systems are the protocol and strategies [LWJ01]. The protocol defines the interaction between the agents. The strategies specify the sequence of actions the agent plans to make

during the negotiation. In CONSENSUS, the protocol (i.e., the negotiation rules) is communicated to the participants before they engage in the negotiation. A human negotiator would consult the protocol; a software agent would download it in a usable format (XML for instance). However, the strategies are the responsibility of the negotiator. A human negotiator would use her strategies to make offers or to respond to her opponents' offers; a software agent would do the same, based on the instructions provided by its creator. In addition to protocols and strategies, we introduce a third component: *coordination*. This is the information that the agents need in order to coordinate their actions while they participate in a combined negotiation (see Section 6.1). Figure 6.1 presents our view of the three components.

Two approaches inspired our representation for the strategies. Su et al. use a high-level rule specification language and GUI tools to allow human negotiation experts to add and change negotiation strategies at run-time [SHH00]. Each strategy is expressed as an Event-Trigger Rule, which specifies the condition to be checked and the actions to be taken by the negotiation server. Rules are activated upon the occurrence of specific events during the negotiation process. Although not directly related to e-negotiations, the work of Grosof et al. on contract representation takes a similar approach [Gro97]: since contract terms usually involve conditional relationships, they are expressed as rules. Similarly, we see strategies as rules that can be fed to the agents before and/or during the negotiation. Strategies such as "*if the bidding gets too intense, then abandon the negotiation*", and "*if you notice any jump-bidding, shielding, or sniping, then wait for further instructions*" can indeed be nicely coded as rules (see Section 6.4).

Rules drive the activity in the agents by describing the action(s) to take when specified conditions are met [MB00]. They are at a relatively high level of abstraction, and are closer to human understanding, especially by domain experts who are typically non-programmers. They are relatively easy to modify dynamically [RGW99], and new behavior can be specified by adding rules, without modifying the previous ones. Furthermore, rules can be augmented with procedural attachments so

that they have an effect beyond pure-belief inferring [GLC99]. Finally, off-the-shelf rule engines can be used to implement rule-based systems. Figure 6.2 shows two rules in the ILOG JRules syntax [ILO02], which implement proxy bidding. Rule1 is triggered when the agent is trailing while its reserve price is not met. In this case it places a bid equal to the actual bid plus the increment. Rule2 enables it to quit whenever its reserve price is met.

```
// If you are not leading and you have not
// reached your reserve price, then bid.
Rule Rule1
{
 When {?x: TheElement(iLead == false;
        (highestBid+myIncrement) <=
         myReservePrice);}
 Then {modify? x
          {action = "BID";}
       }
};
```

```
// If you are not leading and your bid is
// greater than the reserve price, then quit.
Rule Rule2
{
  When {?x: TheElement(iLead == false;
          (highestBid + myIncrement) >
           myReservePrice);}
  Then {modify ?x
            {action = "DROP";}
       }
};
```

Figure 6.2: Proxy bidding in the JRules syntax

6.3 Coordination of Negotiating Agents

6.3.1 Background and Related Work

The interaction between agents can vary from simple information interchanges, to requests for particular actions to be performed and on to cooperation (working together to achieve a common objective) and coordination (arranging for related activities to be performed in a coherent manner) [JFL+01]. Multiple agents need to be coordinated to prevent anarchy; meet global constraints; distribute expertise, resources or information; deal with dependencies between agents; and ensure efficiency [NLJ96]. In CONSENSUS, there is clearly a need to coordinate the software agents (see Section 6.1). First, in a situation where many agents participate in separate negotiations with the goal of purchasing one item (e.g., they engage in many concert ticket auctions with the goal of purchasing one ticket), we need to make sure that only one agent finalizes its deal, and that only the agent in the cheapest auction (i.e., the one with the smallest asking price) is the one to bid. It is common in online auctions to forbid bidders from breaking their commitments. Thus, if more

than one agent end up winning their auctions, they would not be able to retract, or at best they would be, but they would pay a penalty. Second, in a situation where multiple agents negotiate a package of complementary items (i.e., there is no use for one item without the others), we need to make sure that all the items or none are purchased in order to avoid *exposure*. Exposure occurs when we need many items, enter negotiations for all items, and end up making a deal on some but failing to make a deal on others. Coordination in CONSENSUS will be discussed further, but first, we review some related work.

The Biddingbot [IFS+00] is a Multi-Agent System (MAS) that supports users in monitoring multiple auctions. It consists of one leader agent and several bidder agents, each one being assigned to an auction. Bidder agents gather information, monitor, and bid in the multiple sites simultaneously. The leader agent manages cooperation among them, sends user requests to them, and presents bidding information to the user.

Another system, designed at the HP Labs in Bristol, UK, takes a different approach. Instead of a MAS, it uses one single agent [Pri00]. The agent participates in many auctions for one item, and coordinates bids across them using a two-part algorithm: (1) a coordination component, which ensures it has the lowest leading bids; and (2) a belief-based learning and utility analysis component to determine if it should deliberately lose an auction in the hope of doing better in another one later.

Not directly related to e-negotiation research, but highly relevant in the way it deals with coordination in MAS is the ARCHON project [BCL96]. Software agents are divided into two layers: an ARCHON layer and an application program. The former encapsulates knowledge about cooperation, which is domain independent and encoded in terms of production rules. The latter performs the problem solving process. The separation of cooperation knowledge from application knowledge enables generic cooperation features to be reused for other applications.

6.3.2 The CONSENSUS approach

Inspired by the ARCHON approach, we treat coordination information as declarative knowledge, and represent it as if-then rules which the agents exploit using an inference engine. Since our agents are already coupled with rule engines (for their strategy component), it is convenient to use the same rule engine to execute the coordination. We distinguish between strategy rules (described in Section 6.2), used to determine the action to take based on the information about a particular negotiation, and coordination rules, which are used to determine the action to take based on information about other negotiations (possibly combined with information about the negotiation itself). Coordination rules, as well as strategy rules have conditions that are built on the state of the agent, the amount (spent, to spend, committed, remaining, etc.), the remaining time in the negotiation, the frequency of bids, etc.

Suppose we have two agents participating in separate negotiations with the goal of purchasing just one item. The following rule ensures that Agent1 does not commit itself if Agent2 is already committed: "*if Agent2 is leading or is in the process of bidding, then Agent1 should wait.*" Figure 6.3 shows the rule in the JRules syntax. Note that this rule is too restrictive since an agent cannot make a bid if the other one is leading. Evidently, this should not always be the case. If breaking commitments is allowed (perhaps at a certain cost) the rule in question can be relaxed, and we may have the two agents leading at the same time while making sure that one of them drops out of the auction before it is too late.

Suppose now that we have two agents participating in separate negotiations with the goal of purchasing two complementary items. The following rule minimizes the risk of exposure (see Section 3.1): "*if Agent2 is trailing, and its chances of making a deal are slim, then Agent1 should wait for further instructions.*"

```
// If Agent2 is leading, or is in the process of bidding, then Agent1 waits.
Rule coordinate1
{
 Priority = high;
 When {
          ?y: TheElement(iLead == false);
          Blackboard( ?b1: get("u2","iLead"); ?b2:
          get("u2","iBid")) from getBlackboard(); evaluate(?b1 || ?b2);
        }
 Then { modify ?y
          { action = "DO NOTHING";}
          assert logical BidBloc() {ref = new Integer (0);}
      }
};
```

Figure 6.3: A coordination rule in the JRules syntax

Finally, for practical reasons, we adopted a coordination approach where the agents post and read from a general blackboard. Coordination rules are therefore triggered by information that the agents make available in the blackboard (see Figure 6.3). This approach is suitable only if the tasks are assigned, a priori, to the agents, if the number of agents is small, and if the agents share a common domain understanding. Evidently, all requirements are satisfied in our case, otherwise, the blackboard scheme would result in a severe bottleneck.

6.4 Validation

In order to validate our choice of representation for strategies and coordination, we used our agent-mediated negotiation system (i.e., CONSENSUS) to conduct bidding tournaments. The agents function in repeated cycles we call *pulses*. A pulse is made of four *steps* as described by the following piece of code:

```
repeat
sleep (p);
        getInformation ();
        think ();
        act ();
until (state = "winning" or state = "loosing" or state = "dropping")
```

The pulse's steps are:

(1) *sleep (p)*: the agent goes to sleep for a period p that can be fixed or determined dynamically;

(2) *getInformation ()*: the agent queries the server and updates its private information;

(3) *think ()*: the agent applies the rules to determine the action to take; and

(4) *act ()*: the agent takes the appropriate action.

The possible actions are:

(1) *do nothing*: take no action for the time being;

(2) *bid (v)*: bid the amount v; and

(3) *drop*: quit permanently. Finally, the agent's *states* are: trailing, bidding, leading, winning, loosing, and dropping.

Each agent instantiates a rule engine, enabling it to exploit the rules. There are three categories of rules:

(1) basic rules, which determine the agent's behaviour within the negotiation protocol at hand;

(2) strategy rules; and

(3) coordination rules.

Using this setting, we conducted bidding tournaments within a simulated market, and we present here the results involving the English and Dutch auctions, and some coordination schemes across them.

6.4.1 English Auction

In an English auction, the participant has to decide whether or not to bid, how much to bid, and whether or not to abandon the auction (see the actions above). Observing the opponents (as in real life auction houses) is essential in taking such decisions, and by doing so, the participant gains two types of information: (1) how many bidders have dropped out of the auction (since they have lower valuations than the current

bid); and (2) at what prices they dropped out [Mes88]. The participant may also try to predict its opponents' behavior, and sometimes that means guessing their private information. Finally, it might be helpful to know if the opponents are *symmetric* (i.e., they use the same measurements to estimate their valuations), and if they have secret information about the item [Ago02].

Optimal bidding in an English auction means bidding a small amount more than the current bid until you reach your valuation and then stop. This strategy, described by the rules in Figure 6.2, has the effect shown in Figure 6.4. Notice that the time is in seconds, but a real online auction might take a week or more. The winner (Agent in this case) is the one with the highest valuation (i.e., reserve price).

Since online auctions take place over a long period of time, it is hard to monitor them. The reasons are: (1) there is an Internet connectivity cost every time you enter a bid; and (2) there is an opportunity cost associated with logging on and entering the bid. An agent should optimize connections by connecting only when it is relevant to do so, and should be able to determine rapidly whether or not the auction suits its needs. Dropping out early means it can enter another auction early. Figure 6.5 shows Agent adjust its *update rate* to the *average bidding rate*. At the start of the auction, it makes few interventions, and as the activity increases, it participates more often (monitors the auction closely). Note that in this case, the agent makes bids every time it updates its information, but this is not an obligation.

In auctions where no minimum increment is enforced, we talk of *spontaneous bidding*. In this case, it might be useful to observe the bidding process and adapt one's increment to that of the opponents. Figure 6.6 shows Agent doing exactly that.

Jump bidding means entering a bid larger than necessary to send a signal to the opponents. The sender and the recipient of the signal may be better off in a jump bidding *equilibrium*: the sender saves bidding costs by deterring potential competition; and the recipient saves the costs of bidding against a strong opponent [ET99]. Figure 6.7 shows Agent detecting a jump bid made by Player and deciding to quit. Some opponents continue to bid but Player finally wins. In Figure 6.8, Agent

responds with jump bids until Player (and everyone else) quits. In Figure 6.9, Agent detects a jump bid, and waits until the bidding goes back to normal before bidding again.

Sniping means waiting until the last minute of the auction, and trying to submit a bid, which barely beats the high bid and gives the opponents no time to respond. Obviously, if all bidders follow a sniping strategy, the game would become equivalent to a first-price sealed-bid auction, with all the bids submitted at the end. In Figure 6.10, Agent snipes and wins, and in Figure 6.11, three agents engage in a *sniping war.*

Bid shielding happens when a bidder puts in an early low bid (say $10) on an item, and then gets a friend (or a false identity) to put in an extremely high bid (say $500) on the same item. The high bid acts as a shield for the low bid, keeping anyone else from bidding. Just before the end of the auction, the bidder retracts the $500 bid, leaving the $10 bid as the winning bid on an item that should have gone for a higher price. Figure 6.12 shows Player1 and Player2 perform a shielding. Agent quits because the shield exceeded its valuation, and Player1 wins after Player2 retracts its bid. In Figure 6.13, Agent detects the shield and waits for it to be removed to snipe and win.

It can happen that a bidder reconsiders an item's valuation by observing how the opponents behave. Figure 6.14 shows Agent increase its valuation when its reserve price is met, the auction is about to close, and few participants remain. It changes the reserve price from 600$ to 900$ and stays in the game to win.

Finally, Figure 6.15 shows a combination of a several tactics. Agent avoids jump bidding by entering a waiting state, and at the end, it snipes and wins. This could be a way to hide your real intentions from your opponents.

6.4.2 Dutch Auction

When the market price of the auctioned item is a *common valuation* among bidders (e.g., an auction for personal computers), a bidder who wins the auction is the one

with the highest yet possibly overrated valuation. This is called the *winner's curse.* In Dutch auctions, where a bidder selects a cut-off price at which to claim the item so long as no one else claims it, all participants try to avoid the winner's curse by *shading down* their bids slightly below their valuations (i.e., bidding less than what they think the object is worth) [Mes88].

In Dutch auctions, no relevant information is disclosed in the course of the auction, only at the end when it is too late, which makes designing strategies a futile task. This is not the case with *multi-item Dutch auctions* where bidders watch the transaction prices, the number of remaining items (if available), the remaining time (if available), and decide whether to bid, wait for the price to drop, or quit. We designed a multi-item Dutch auction as follows: (1) identical items are put on sale and the unit price is made public, (2) as time passes, the seller decreases the unit price to generate interest, and (3) a buyer's bid is the quantity to purchase at the current price. The basic behavior of our agents is not different from that of the English auction. Instead of bidding a price, the agents bid a quantity. When created, an agent is given the quantity to buy, the minimum acceptable quantity, and the user's valuation.

We define a "safe buying" tactic as: "*as the price decreases, when it reaches your valuation, buy the minimum quantity. Keep watching the auction, and before it closes, buy the remaining items (possibly at a smaller price that your valuation).*" The effect of this tactic is shown in Figure 6.16, where Agent (right) buys 4 items (minimum), and later, buys 3 more items to reach its maximum quantity of 7.

We define "panic buying" tactic as: "*as the price decreases, if you see that a large number of items are sold, and your valuation is not met, then it might mean that your valuation is too low. Adjusting your valuation up will permit you to buy at least the minimum quantity.*" In Figure 6.17, Agent (right) increases its valuation and buys the minimum 4 items needed. Notice that the risk of missing the minimum quantity is small, but the risk of the winner's curse is high.

Finally, Figure 6.18 shows the effect of a the following "patient buying" tactic: "*if the price drops fast, the buying rate is low, and you reach your valuation, then lower*

your valuation". This tactic might save the buyer from the winner's curse as it helps get the items at a lower price. There is however a risk that someone else buys all the items before the agent makes its move.

6.4.3 Coordination

Using the same setting as before, we made several agents participate in separate auctions at the same time, and used the rule-based approach to manage their coordination. Here are some results.

Suppose we want two complementary items A and B, and we engage Agent1 in an English auction for A, and Agent2 in another English auction for B. The two agents negotiate separately, but they coordinate their actions using rules. Figure 6.19 highlights the following coordination scheme: "*if Agent1 (left) detects a jump bid (i.e., the opponent is serious about winning, therefore Agent1 may loose its auction) then Agent1 must quit. In this case, Agent2 (right) must also quit to avoid exposure.*"

Now, suppose we want item A or item B, and we launch Agent1 in an English auction for A, and Agent2 in a Dutch auction for B. Figure 6.20 shows the effect of the following two coordination rules: "*if the going price in the Dutch auction (right) is less than the current bid in the English auction (left), and our valuation is higher than the going price, we buy in the Dutch auction*", "*if the going price in the Dutch auction is less than the current bid in the English Auction, we quit the English auction.*"

For the following tests, three agents Agent1, Agent2, and Agent3 participate at the same time in separate (and independent) English auctions for items B, C and A respectively. The goal is to win "(B or C) and A", that is "B and A" or "C and A". Figure 6.21 shows Agent1 and Agent2 (left) bidding in parallel, and only one is leading at the same time while bids are always made in the cheapest auction. The figure also shows Agent3 (right) quitting (its reserve price is met), causing Agent1 and Agent2 (left) to quit. In Figure 6.22, both Agent1 and Agent2 (left) loose (their reserve price is met). At the same time, Agent3 (right) quits. Figure 6.23 shows

Agent3 (right) snipe and win, and Agent1 and Agent2 (left) continue until Agent2 wins. In Figure 6.24, Agent3 (right) engages in a sniping war and looses to another sniper. Agent1 and Agent2 (left) have no choice but to quit.

6.5 Conclusion

The aim of the paper was to demonstrate that we could capture a wide range of bidding strategies and bid coordination schemes, using a rule-based approach, while supporting a wide spectrum of negotiation types. The well-known advantages of rule-based systems are the modularity and the uniformity (knowledge is represented using the same format). The possible inefficiency of rules and their eventual slow processing can be overcome by compiling the rules. Graphical tools can also be used to browse the rule base in order to verify its coherence. Bid strategies usually involve fairly complex optimization algorithms and forecasting that could not be expressed directly as rules. We propose to make use of optimization algorithms and forecasting as procedural attachments in the action part of the rules.

Basic behavior of agents, various bidding tactics and coordination schemes across multiple auctions were implemented using our representation. They were tested in agent tournaments within simulated markets. So far, the results are encouraging and other possibilities are yet to be explored. As further points of interest we see the following. (1) The flexibility of this representation should be tested, and the limits of its expressiveness should be sought. (2) It is no secret that rules are a bit restricted in their ability to be adapted automatically, as agents are usually expected to adapt over time to improve their performance. It should therefore be investigated whether our approach can lend itself to machine learning techniques [WZK00, ZS98]. (3) We have been considering negotiations where the only negotiable attribute is the price, but in the case of a plane ticket, for instance, the attributes could be the price, the date of the flight, the airports (departure and destination), etc. Coordination across several multi-attribute negotiations should therefore be studied since it generates more complexity.

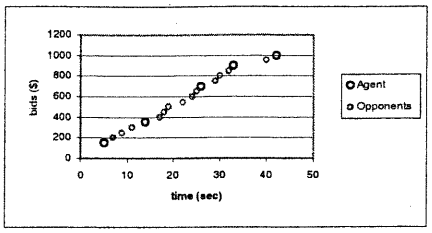

Figure 6.4: Optimal bidding

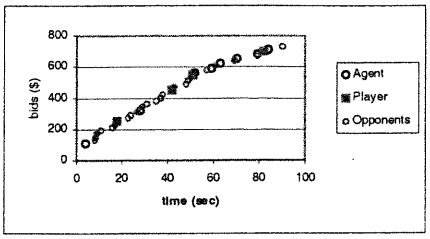

Figure 6.5: Adjust update rate

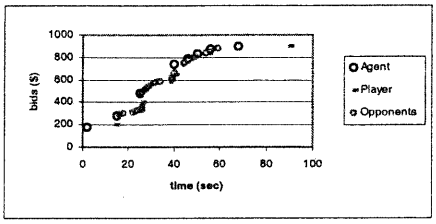

Figure 6.6: Adapt increment

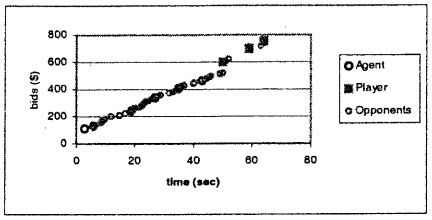

Figure 6.7: Jump bid – detect and quit

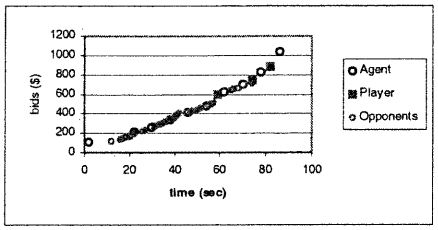

Figure 6.8: Jump bid – detect and respond

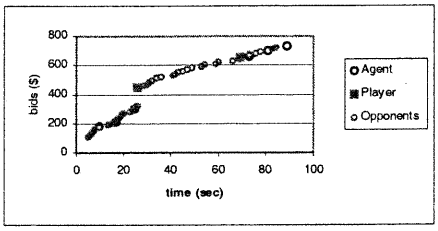

Figure 6.9: Jump bid – detect and wait

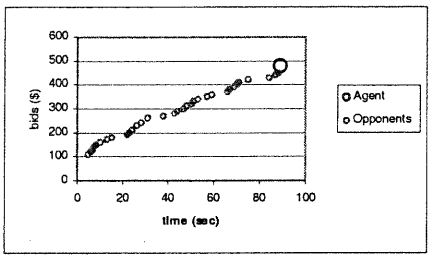

Figure 6.10: Snipe and win

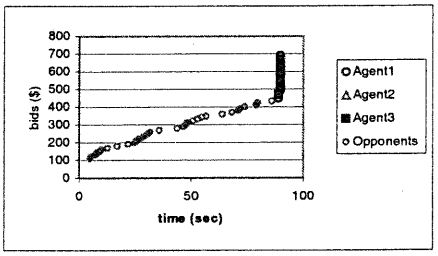

Figure 6.11: Sniping war

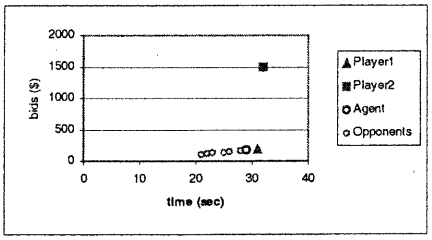

Figure 6.12: Shielding – detect and drop

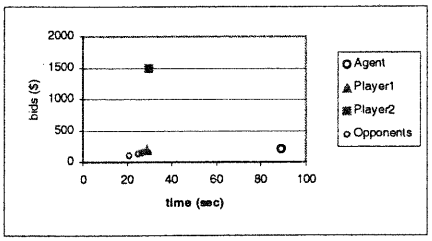

Figure 6.13: Shielding – detect and snipe

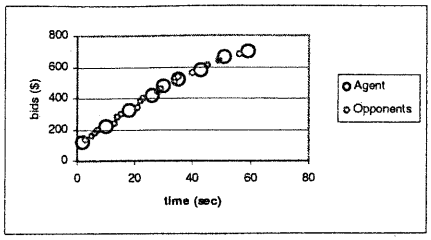

Figure 6.14: Increase valuation

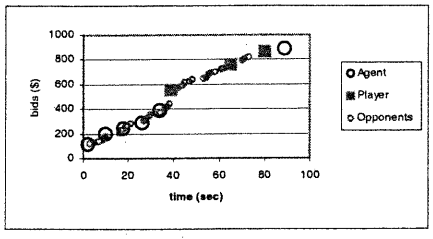

Figure 6.15: Jump bid - detect, wait & snipe

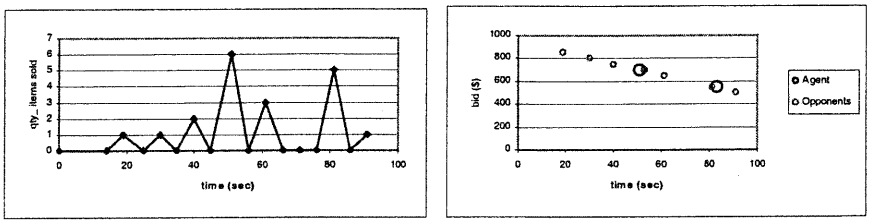

Figure 6.16: Multi-item Dutch – safe buying

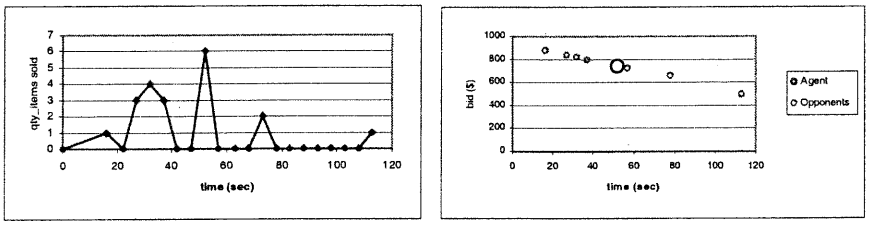

Figure 6.17: Multi-item Dutch – panic buying

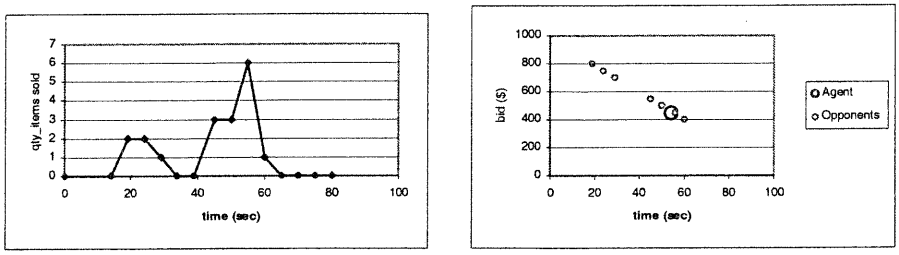

Figure 6.18: Multi-item Dutch – patient buying

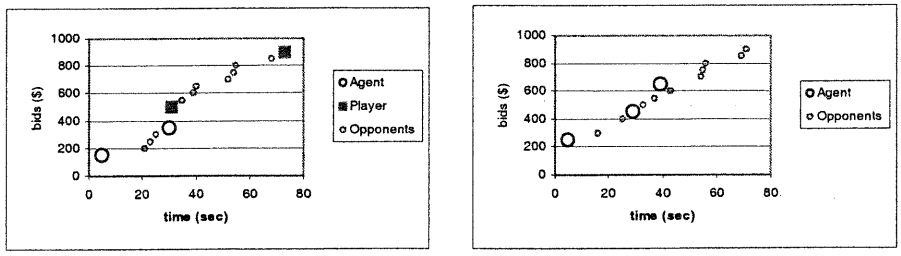

Figure 6.19: Coordination – two English auctions

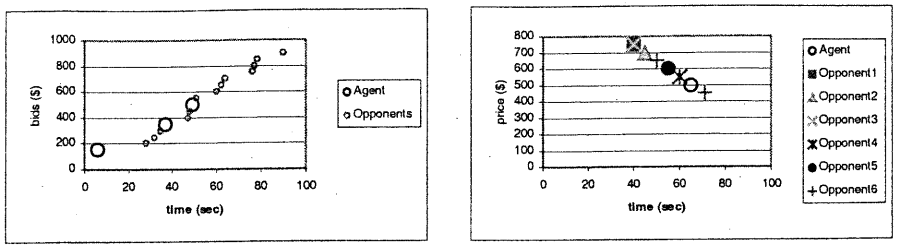

Figure 6.20: Coordination – English and Dutch auctions

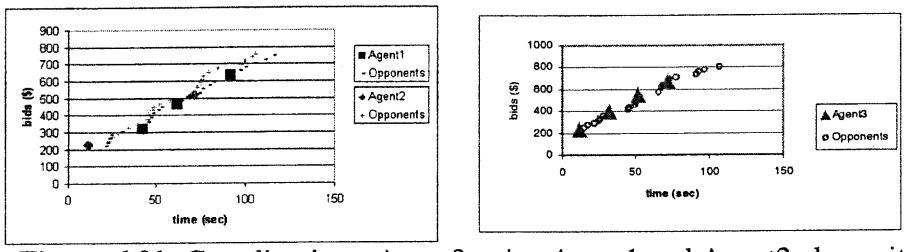

Figure 6.21: Coordination - Agent3 quits, Agent1 and Agent2 also quit

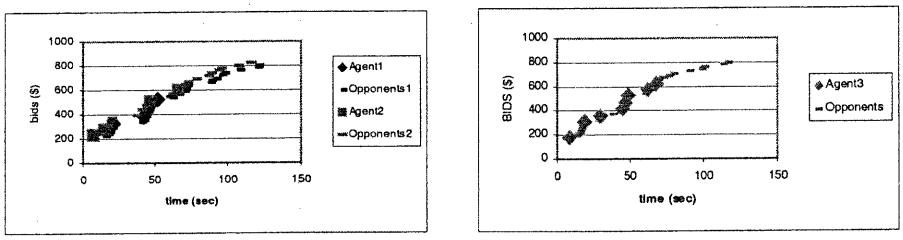

Figure 6.22: Coordination - Agent1 and Agent2 loose, Agent3 quits

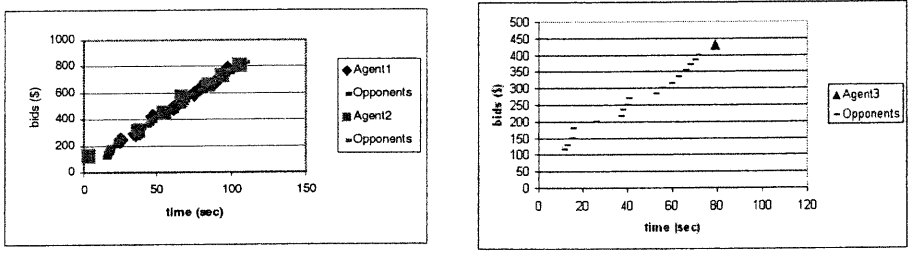

Figure 6.23: Coordination - Agent3 snipes and wins, Agents1 and Agent2 keep going

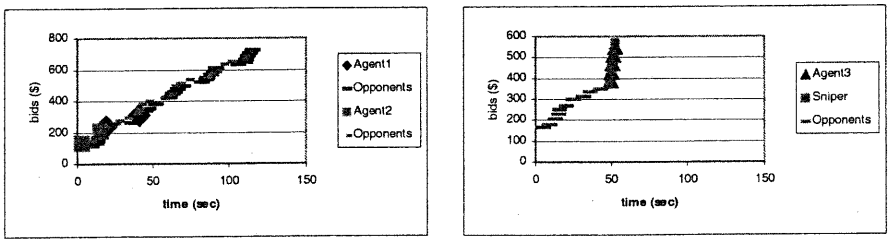

Figure 6.24: Coordination - Agent3 snipes and looses, Agent1 and Agents2 quit

CHAPTER 7

CONCLUSION

In this chapter, we present an assessment of our work, we review the major contributions, and detail the points that need to be addressed as future work.

7.1 Assessment

A combined negotiation (CN) involves negotiating several interdependent goods in separate and independent negotiations. We introduced and defined this new negotiation type, pointed out its challenges, identified the research issues it raises, and suggested a software solution to conduct it. The solution is a combined negotiation support system (CNSS) that a human consumer can use according to the following e-commerce negotiated transaction scenario: (1) a discovery phase where the consumer uses the system to find the goods and the providers of the goods; (2) a modeling phase where she uses the system to specify the sequencing of the negotiations and the dependencies between them, as well as to enter the negotiation strategies and coordination information; and (3) an enactment phase where the consumer runs the model, and possibly intervenes in it whenever necessary.

With that in mind, we designed a complete architecture for a CNSS we call CONSENSUS. The architecture had to be: (1) realistic so that we could implement it; (2) based on reuse and off-the-shelf components so that we would finish its implementation in time to evaluate it, (3) modular and resilient to evolving functionality so that several people could participate in implementing it, and prototypes could be used immediately to validate various functionalities individually; and finally (4) flexible enough to be used as a research infrastructure. The experience

reported in Chapter 4 [BAV+01] makes us believe we succeeded in achieving these goals.

The driving idea behind our solution is: (1) a workflow that captures the sequencing and the control flow of the CN, i.e., the CN know-how; (2) software agents that carry out the individual negotiations, i.e., individual negotiation know-how; (3) strategy rules to help the agents decide what to do when there are many options to choose from, i.e., negotiation strategy know-how; and (4) coordination rules to manage the agents across several negotiations, i.e., coordination know how. Workflow technology brings a high level of reliability; software agents bring automation; and if-then rules provide modularity, uniformity, a high level of abstraction, versatility, ease of use, and the possibility to modify the knowledge dynamically.

An important challenge was to allocate functionalities to the four components (workflow, agents, strategies, and coordination) in a clean an optimal way. Again, we believe we succeeded in doing so [BAV+01].

The implementation of the architecture was incremental, and four people participated in it, leading to CONSENSUS version 1.0[30]. It is clear that our architectural choices and characteristics played an important role in making the implementation work efficient and smooth.

Our experience with GNP proved to be positive. The generic qualities of GNP made it possible for a non-programmer (i.e., an economist designing a new negotiation protocol) to script a new negotiation within a reasonable time frame, and without an excessive effort. Several auction types such as the English, Dutch, and multi-item Dutch were implemented and used within CONSENSUS.

We used CONSENSUS and GNP to model and enact several simple and complex CN cases. We were encouraged by the results, although many improvements are still needed to make the solution easier to deploy and use by a non-initiated consumer.

---

[30] Work is in progress to produce the next version.

We used the same settings as before to conduct bidding agent tournaments. The goal was to show that we could rely on a declarative representation to provide software agents with simple bidding tactics and simple coordination schemes. The tournaments were not meant to evaluate tactics or to show that one tactic is more efficient than another one. The goal was rather to show that tactics could be implemented as rules. The results were encouraging (see Chapter 6 [BAK02]), and the experimentation settings can be used for further investigations regarding the representation mechanism itself, or for the evaluation of known and newly designed negotiation strategies.

7.2 Contributions

We divided our major contributions into three categories: conceptual, architectural, and general.

Conceptual contributions

In this thesis, we set the basis for a new negotiation type: CNs. We also identified the issues CNs generate, and their level of complexity. We demonstrated the need for a CNSS to conduct them both at the B2C and the B2B level, and defined the requirements for such a tool. We then devised a complete conceptual architecture for a CNSS we call CONSENSUS.

We demonstrated that we could capture a wide range of bidding strategies and bid coordination schemes, using a rule-based approach, while supporting a wide spectrum of negotiation types.

We demonstrated the need for a formal description of negotiation rules, and suggested UML statecharts as a mechanism to respond to this need. This enabled us to understand, implement and test different negotiation protocols more thoroughly before using them as nodes in a CN.

We participated in the drafting of a "negotiation taxonomy," as part of our involvement and collaboration with the E-negotiations Group.

We showed the need for dynamism in CNs and pointed to the lack of flexibility in static WfMS [BBK+02].

Finally, a CN eventually leads to a set of contracts between the consumer and the different providers of the items (goods and services) that make the package. These contracts are possibly interrelated, the same way the negotiations that lead to them are. Electronic contracting (e-contracting) aims at providing IT support for all the phases of a contract lifecycle. Two important phases are the drafting of the contract and its fulfillment. Multiple interrelated contracts certainly bring another level of complexity to e-contracting research. This research work is being pursued at the moment as part of an internship at IBM T J Watson Laboratory in Hawthorne, New York.

Architectural contributions

We experimented with a novel way of using workflows to model CNs at build time, and to enact and monitor them at runtime, and used software agents as actors in the workflow. We experimented with a declarative, rule-based representation of strategies and coordination schemes. The basic behavior of software agents, various bidding tactics, and coordination schemes across multiple auctions were implemented using our representation. They were tested in agent tournaments within simulated markets.

We contributed to the design and later to the validation of GNP by using instances of it as negotiation servers in order to run the negotiations that made up the nodes of a CN.

The proposed architecture was incrementally validated by proof-of-concept implementations. A stable implementation of the architecture of CONSENSUS (i.e., version 1.0) is available.

General contributions

TEM is a large an ambitious project, and Combined Negotiations are just one facet of it. Other directions are being taken, mostly in auction design, economic experiments of new auction models, operations research applied to transportation, etc. This research brought modest visibility to TEM in the e-negotiations community through

active participation in e-negotiations workshops, and e-commerce conferences. CONSENSUS was also a winner at the 2000 IAC[31] contest for best thesis proposal in e-commerce.

7.3 Research Perspectives

The possibility to instantiate software agents from a formalized description of the negotiation rules should be investigated. The same description could be used to generate automatically the script that implements the negotiation on GNP, thus making GNP an even more generic negotiation platform.

A High-level CN Specification Tool is yet to be implemented. It should enable the user to visually specify the CN, and a workflow generator should be used to transform the specification into a workflow. As the user of CONSENSUS might or might not be a domain specialist, a library of domain-specific workflows should be deployed.

The question of breaking commitments that a participant makes during the negotiation process should be investigated along with its effect on CNs. This calls for designing and implementing protocols that allow for commitments to be broken. GNP is well suited for supporting such protocols, and when included as nodes in CNs, these protocols may be studied using CONSENSUS.

The study of the cooperative behavior of software agents in conducting a CN looks promising, and we could rely on GNP and CONSENSUS to pursue it. Taking a similar approach, the agents' competitive behavior could be studied.

Integrating a dynamic WfMS into the current prototype of CONSENSUS will enable user intervention through the Workflow Monitoring and Control Tool of the current CONSENSUS architecture. A wish list of dynamic modifications required for CNs should be produced. We need to be aware of the shortcomings and limitations of current dynamic WfMSs in respect to the modeling and running of CNs. These

---

[31] Institute for Advanced Commerce. IBM TJ Watson research Lab. http://www.research.ibm.com/iac/

limitations should give us a valuable input for future versions of current dynamic workflow systems.

Regarding the rule-based representation of strategies and coordination, the following points should be investigated:

- The representation's flexibility should be further tested, and the limits of its expressiveness should be explored.
- It is no secret that rules are somewhat restricted in their ability to be adapted automatically. However, agents are usually expected to adapt over time to improve their performance. It should therefore be investigated whether our approach can lend itself to machine learning techniques.
- We have been considering negotiations where the only negotiable attribute is the price, but in the case of a plane ticket, for instance, the attributes could be the price, the date of the flight, the airports (departure and destination), etc. Coordination across several multi-attribute negotiations should therefore be studied and complexity should be addressed.

Finally, we should leverage web services and adapt the CONSENSUS architecture. To this end, we suggest the following:

- Replacing the "Product/Merchant Brokering System" with a "Discovery Component" based on UDDI [UDD00]. This component will discover the products and the providers of the items making the package that is the object of the CN.
- Binding to the negotiations using WSDL [WSD01] and SOAP [SOA00].
- The sequencing of the individual negotiations that make up the CN could be realized using WSFL [WSF01].

BIBLIOGRAPHY

[Ago02] Agorics. http://www.agorics.com/new.html.

[Ama02] Amazon Website. http://www.amazon.com.

[AME00] AMEC Laboratory. http://ecommerce.media.mit.edu/.

[Auc00] The Michigan Internet Auctionbot. http://auction.eecs.umich.edu/.

[BAK01] Morad Benyoucef, Hakim Alj, and Rudolf K. Keller. An Infrastructure for Rule-Driven Negotiating Software Agents. In Proceedings of the Twelfth International Workshop on Database and Expert Systems Applications (DEXA 2001), pages 737-741, Munich, Germany, September 2001. IEEE. Presented in 2nd e-Negotiations Workshop.

[BAK02] Morad Benyoucef, Hakim Alj, Kim Levy, and Rudolf K. Keller. A Rule-driven Approach for Defining the Behavior of Negotiating Software Agents. In Proceedings of the Fourth International Conference on Distributed Communities on the Web, Sydney, Australia, April 2002. Springer. LNCS. To appear.

[Bar00] Bargainfinder Website. http://www.bargainfinder.com.

[BAV+01] Morad Benyoucef, Hakim Alj, Mathieu Vézeau, and Rudolf K. Keller. Combined Negotiations in E-Commerce: Concepts and Architecture. Electronic Commerce Research Journal, 1(3):277-299, July 2001. Special issue on Theory and Application of Electronic Market Design. Baltzer Science Publishers.

[BB00] A. Amit Basu and R.W. Blanning. A Formal Approach to Workflow Analysis. Information Systems Research, 11(1), pages 17-36, March 2000.

[BBK01] Morad Benyoucef, Sarita Bassil and Rudolf K. Keller. Workflow Modeling of Combined Negotiations in E-Commerce. In Proceedings of The Fourth International Conference on Electronic Commerce Research (ICECR-4), pages 348-359, Dallas, TX, USA, November 2001.

[BBK+02] Sarita Bassil, Morad Benyoucef, Rudolf K. Keller, and Peter G. Kropf. Addressing Dynamism in E-Negotiations by Workflow Management Systems. In Proceedings of the Thirteenth International Workshop on Database and Expert Systems Applications (DEXA 2002), Aix-en-Provence, France, September 2002. IEEE. Presented in 3rd e-Negotiations Workshop. To appear.

[BCL96] M Berndtsson, S. Chakravarthy, and B. Lings. Coordination among agents using reactive rules. Technical Report HS-IDA-TR-96-011, Skovde University, Sweden, October 1996.

[BEA02] BEA WebLogic Process Integrator. http://www.beasys.com/products/weblogic/integrator/.

[Ben00] Morad Benyoucef. Support for combined E-negotiations: Concepts, architecture, and evaluation. Technical Report GELO-122, Université de Montréal, Montréal, Québec, Canada, May 2000. Thesis proposal.

[Bic00] Martin Bichler. A Roadmap to Auction-based Negotiation Protocols for Electronic Commerce. In proceedings of the 3rd Hawaii International Conference on Systems Sciences. 2000.

[BK00a] Morad Benyoucef and Rudolf K. Keller. An Evaluation of Formalisms for Negotiations in E-Commerce. In Proceedings of the Workshop on Distributed Communities on the Web, pages 45-54, Quebec City, QC, Canada, June 2000. Springer. LNCS 1830.

[BK00b] Morad Benyoucef and Rudolf K. Keller. A Conceptual Architecture for a Combined Negotiation Support System. In Proceedings of the Eleventh International Workshop on Database and Expert Systems Applications (DEXA 2000), pages 1015-1019, Greenwich, London, Britain, September 2000. IEEE. Presented in W17: Workshop on Negotiations in Electronic Markets - Beyond Price Discovery - E-Negotiations.

[BK00c] Morad Benyoucef and Rudolf K. Keller. A conceptual architecture for a combined negotiation support system. Technical Report GELO-118, Montreal, Canada, February 2000.

[BK99] Morad Benyoucef and Rudolf K. Keller. A survey on description techniques for auction rules in e-commerce. Technical Report GELO-101, Montreal, Quebec, Canada, August 1999.

[BKL+00] Morad Benyoucef, Rudolf K. Keller, Sophie Lamouroux, Jacques Robert, and Vincent Trussart. Towards a Generic E-Negotiation Platform. In Proceedings of the Sixth International Conference on Re-Technologies for Information Systems, pages 95-109, Zurich, Switzerland, February 2000. Austrian Computer Society.

[BS97] Carrie Beam and Arie Segev. Automated negotiations: A survey of the state of the art. Technical Report 97-WO-1022, Haas School of Business, UC Berkeley, 1997.

[BSS96] Carrie Beam, Arie Segev, and J. George Shanthikumar. Electronic negotiation through internet-based auctions. Tech. Rep. 96-WP1019, Haas School of Business, UC Berkeley, December 1996.

[CHR+98] A. Cichocki, A. Helal, M. Rusinkiewicz, and D. Woelk. Workflow and Process Automation: Concepts and Technology, Kluwer 1998.

[CKO92] B. Curtis, M. I. Kellner and J. Oliver. Process Modeling. Communications of the ACM. Vol. 35, No. 9, pages 75-90, September 1992.

[CLD99] Peter Coad, Eric Lefebvre, and Jeff De Lucas. Java Modeling in Color with UML. Prentice-Hall, 1999.

[CLL+99] Aaron G. Cass, Hyungwon Lee, Barbara Staudt-Lerner, and Leon J. Osterweil. Formally defining coordination processes to support contract negotiation. Technical Report UM-CS-1999-039, University of Massachusetts, Amherst, MA, June 1999.

[CM96] Anthony Chavez and Pattie Maes. Kasbah: An agent marketplace for buying and selling goods. In The First International Conference on the Practical Application of Intelligent Agents and Multi-agent Technology, pages 75-90, London, UK, April 1996.

[DOM00] Document Object Model (DOM) level 1 specification, version 1.0. http://www.w3.org/TR/REC-DOM-Level-1/.

[EAu02] The eAuctionHouse Site. http://ecommerce.cs.wustl.edu/eAuctionHouse/. 2002.

[Eba02] EBay Auctions. http://www.ebay.com.

[ECD97] ECDTF Reference Model. Technical report. Object Management Group (OMG), 1997. Consulted at http://www.oms.net/ecdtf.html.

[EJB00] Enterprise JavaBeans Specifications. http://java.sun.com/products/ejb/docs10.html.

[ENe02] The ENegotiations Group Web Page. http://enegotiations.wu-wien.ac.at/.

[ET99] Robert F. Easley and Rafael Tenorio. Bidding Strategies in Internet Yankee Auctions. Working paper, University of Notre Dame, 1999.

[GGR98] Eduard Giménez-Funes, Lluís Godo, Juan A. Rodríguez-Aguilar and Pere Garcia-Calvés. Designing bidding strategies for trading agents in electronic commerce. In Third Intl Conference on Multi-Agents Systems (ICMAS-98), pages 136-143, Paris, France, July 1998.

[GK99] Amy R. Greenwald and Jeffrey O. Kephart. Shopbots and pricebots. In16th Intl. Joint Conference on AI, volume 1, pages 506-511, Stockholm, Sweden, August 1999.

[GLC99] Benjamin N. Grosof, Yannis Labrou, and Hoy Y. Chan. A declarative approach to business rules in contracts: Courteous logic programs in XML. In 1st Conference on Electronic Commerce, Denver, Colorado, November 1999.

[GMM98] Robert H. Guttman, Alexandros G. Moukas, and Pattie Maes. Agent-mediated electronic commerce: A survey. Knowledge Engineering Review, 13(3), June 1998.

[Gro97] Benjamin N. Grosof. Courteous logic programs: Prioritized conflict handling for rules. Technical Report RC 20836, IBM Research, T.J. Watson Research Center, May 1997.

[HH99] Daniela Handl and Hans-Juergen Hoffmann. Workflow Agents in the Document-centered communication in MALL2000 systems. In Proceedings of the International Workshop on Agent-Oriented Information Systems (AOIS™99). Seattle, WA, May 1999.

[HRW99] Junling Hu, Daniel Reeves, and Hock-Shan Wong. Agent service for online auctions. In Workshop on Artificial Intelligence in Electronic Commerce, Menlo Park, CA, 1999.

[HG96] D. Harel and E. Gery. Executable object modeling with statecharts. In Proc. of 18th Intl. Conference on Software Engineering, pages 246-257, Berlin, Germany, March 1996.

[HLN+90] D. Harel, H. Lachover, A. Naamad, A. Pnueli, M. Politi, R. Sherman, A. ShtullTrauring, and M. Trakhtenbrot. STATEMATE: A working environment for the development of complex reactive systems. IEEE Transactions on Software Engineering, 16(4):403-414, 1990.

[Hur00] Hurwitz Report. Negotiated Trade: the Next Frontier for B2B e-commerce. Technical Report. 2000.

[IBM02] IBM MQSeries Workflow. http://www-4.ibm.com/software/ts/mqseries/workflow/.

[IFS+00] Takayuki Ito, Naoki Fukuta, Toramatsu Shintani and Katia Sycara. Biddingbot: A multi-agent support system for cooperative bidding in multiple auctions. In 4th Intl Conference on Multi-agent Systems (ICMAS-2000), Boston. MA, July 2000.

[ILO02] ILOG JRules. http://www.ilog.com/products/jrules/.

[JB96] Stefan Jablonski and Christoph Bussler. Workflow Management, Modeling Concepts, Architecture, and Implementation. International Thomson Computer Press, 1996.

[JFL+01] N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge. Automated negotiation: Prospects, methods and challenges. International Journal of Group Decision and Negotiation, 10(2), 2001. To appear.

[Jhi99] Anant Jhingran. The Emergence of Electronic Market Places, and other E-Commerce Directions. Handout at Workshop on Electronic Marketplaces held at Cascon'99, Toronto, ON, Canada, November 1999.

[Jon99] Kees Jonkheer. Intelligent Agents, Markets and Competition: consumers' interests and functionality of destination sites. Electronic publication. http://www.firstmonday.com. 1999.

[Jpy02] The JPython Language. http://www.jpython.org.

[JSP00] Java Server Pages Specifications. http://java.sun.com/products/jsp/techinfo.html.

[Ker99] Gregory E. Kersten. Decision Making and Decision Support. In Decision Support Systems for Sustainable Development in Developing Countries, pages 29-51, Kluwer Academic, 1999.

[KF98a] Manoj Kumar and Stuart I. Feldman. Business negotiations on the Internet. In INET98 Conference of the Internet Society, Geneva, Switzerland, July 1998.

[KF98b] Manoj Kumar and Stuart I. Feldman. Internet auctions. Technical report, IBM Institute for Advanced Commerce, Yorktown Heights, NY, November 1998.

[KHG00] Jeffrey O. Kephart, James E. Hanson and Amy R. Greenwald. Dynamic pricing by software agents. Computer Networks, vol.32, no.6, Elsevier, pages 731-752, May 2000.

[KL01] Gregory E. Kersten and Gordon Lo. Negotiation Support Systems and Software Agents in E-business Negotiations. Technical Report INR05/01. The InterNeg Group. Concordia and Carleton Universities, Canada. 2001.

[KNT02] Gregory E. Kersten, Sunil J. Noronha, and Jeffrey Teich. Control architecture for a flexible Internet auction server. In Fourth International Conference on the Design of Cooperative Systems, Sophia-Antipolis, France, May 2000. To appear.

[KS98] Gregory E. Kersten and S. Szpakowicz. 7th Workshop on Intelligent Information Systems. Collocated with Fourth International Conference on the Design of Cooperative Systems, Warsaw, Poland, 1998.

[KT00] Gregory E. Kersten and Jeffrey Teich. Are all E-Commerce Negotiations Auctions? In Fourth International Conference on the Design of Cooperative Systems, Sophia-Antipolis, France, May 2000.

[KT01] Pinar Keskinocak and Sridhar Tayur. Quantitative Analysis for Internet-Enabled Supply Chains. INTERFACES 31:2 pages 70-89. March-April 2001.

[KW82] David M. Kreps and Robert Wilson. Sequential equilibria. Econometrica, 50(4):863–894, July 1982.

[LG99] Sophie Lamouroux and Robert Gérin-Lajoie. Le serveur de négociation électronique GAMME. Technical report, CIRANO, Montreal, Canada, 1999.

[LK99] Gordon Lo and Gregory E. Kersten. Negotiation in Electronic Commerce: Integrating Negotiation Support and Software Agent Technology. In Proceedings (CD-ROM) of the 29th Atlantic Schools of Business Conference, Halifax, NS, 1999.

[LS97] Yu Lei and Munindar P. Singh. A Comparison of Workflow Metamodels. In Proceedings of the ER-97 Workshop on Behavioral Modeling and Design Transformations: Issues and Opportunities in Conceptual Modeling, Los Angeles. November 1997.

[LS98] Herman Lam and Stanley Su. Component Interoperability in a Virtual Enterprise Using Events/Triggers/Rules. In Proceedings of OOPSLA '98 Workshop on Objects, Components, and Virtual Enterprise, October 18-22, 1998, Vancouver, BC, Canada.

[Luc99] David Lucking-Reiley. Auctions on the Internet: What's being auctioned and how? Journal of Industrial Economics. Vol. 48, No. 3, pages 227-252.

[LW99] Heiko Ludwig and Keith Whittingham. Virtual enterprise coordinator - agreement-driven gateways for cross-organizational workflow management, pages 29–38, San Francisco, CA, 1999.

[LWJ01] Alessio Lomuscio, Michael Wooldridge and Nicholas Jennings. A classification Scheme for negotiation in electronic commerce. In Agent-mediated e-commerce: a European AgentLink Perspective, pages 19-33, Springer-Verlag, 2001.

[Mal87] Malone, T.W., et al., Electronic Markets and Electronic Hierarchies. Communications of the ACM, 1987, 30(6): p.483-494.

[MB00] Colleen McClintock and Carole Ann Berlioz. Implementing Business Rules in Java. In Java Developers Journal, pages 8-6, May 2000.

[Mes88] Loretta J. Mester. Going, Going, Gone: Setting Prices with Auctions. Federal Reserve Bank of Philadelphia Business Review (March/April):3-13 1988

[MGM98] Alexander Moukas, Robert Guttman, and Pattie Maes. Agent-mediated Electronic Commerce: An MIT media laboratory perspective. In International Conference On Electronic Commerce, ICEC'98, Seoul, Korea, April 1998.

[MGM99] Pattie Maes, Robert H. Guttman, and Alexandros G. Moukas. Agents that buy and sell: Transforming Commerce as we know it. Communications of the ACM, 42(3):81-91, March 1999.

[MM86] Preston McAfee and John McMillan. Analyzing the Airwaves Auction. In Journal of Economic Perspectives, 10(1), pages 159-175, 1996.

[MM87] Preston McAfee and John McMillan. Auctions and Bidding. Journal of Economic Literature, 25:699-738, 1987.

[MWG95] Andreu Mas-Collel, Michael Whinston, and Jerry Green. Microeconomic Theory. Oxford University Press, 1995.

[MWW98] Peter Muth, Jeanine Weissenfels and Gerhard Weikum. What workflow technology can do for electronic commerce? In EURO-MED NET Conference, Nicosia, Cyprus, March 1998.

[NBB+02] Dirk Neumann, Morad Benyoucef, Sarita Bassil and Julie Vachon. A Comparison of Electronic negotiation Systems. Submitted to Group Decision and Negotiation, Special issue on e-negotiations. 2002.

[NLJ96] H. S. Nwana, L. Lee, and N. R. Jennings. Co-ordination in software agent systems. BT Technology Journal, 14(4):79-89, October 1996.

[Oli97] Jim R. Oliver. Artificial agents learn policies for multi-issue negotiation. International Journal of Electronic Commerce, 1(4):49-88, 1997

[OMG99] Object Management Group (OMG) Negotiation Facility final revised submission. Technical report. March 1999.

[Ons00] Onsale Auctions. http://www.onsale.com.

[Par99] David Parkes. iBundle: an efficient ascending price bundle auction. In proceedings of the first ACM EC conference. Denver Colorado, USA. 1999.

[Per00] Personalogic Website. http://www.personalogic.com.

[Pri00] Chris Priest. Algorithm design for agents which participate in multiple simultaneous auctions. Tech. Report HLP-2000-88, Hewlett-Packard, Bristol, England, July 2000.

[PUF99] D. C. Parkes, L. H. Ungar, and D. P. Forster. Agent Mediated Electronic Commerce, chapter Accounting for Cognitive Costs in On-line Auction Design. In Agent Mediated Electronic Commerce, (LNAI 1571), pages 25-40. Springer, 1999.

[RGW+99a] Daniel M. Reeves, Benjamin N. Grosof, Michael P. Wellman and Hoi Y. Chan. Toward a declarative language for negotiating executable contracts. In Workshop on AI in Electronic Commerce, Menlo Park, CA, 1999.

[RGW+99b] Daniel M. Reeves, Benjamim N. Grosof, Michael P. Wellman, and Hoi Y. Chan. Automated negotiations from formal contract descriptions. In IBM/IAC Workshop on Internet-based Negotiation Technologies, March 1999.

[RJB99] Jim Rumbaugh, Ivar Jacobson, and Grady Booch. The UML Reference Manual. Addison-Wesley, 1999.

[RMN+98] Juan A. Rodriguez-Aguilar, Francisco J. Martin, Pablo Noriega, Pere Garcia and Carles Sierra. Towards a test-bed for trading agents in electronic auction markets. AI Communications, 11(1):5-19, 1998.

[RS97] A. Rangaswamy and G. R. Shell, Using Computers to Realize Joint Gains in Negotiations: Toward an Electronic Bargaining Table. Management Science, vol. 43, pages 1147-1163, 1997.

[San99a] Tuomas Sandholm. Automated negotiation. Communications of the ACM, 42(3):84-85, March 1999.

[San99b] Tuomas Sandholm. An Algorithm for Optimal Winner Determination in Combinatorial Auctions. In International Joint Conference on Artificial Intelligence, pages 542-547, Stockholm, Sweden, 1999.

[SHH00] Stanley Y. W. Su, Chunbo Huang and Joachim Hammer. A replicable web-based negotiation server for e-commerce. In Proc. (CD-ROM) of 33rd Intl. Conf. on System Sciences, Hawaii, 2000.

[SL95] Tuomas Sandholm and Victor Lesser. Issues in Automated Negotiation and Electronic Commerce: Extending the Contract Net Framework. In International Conference on Multi-Agent Systems, pages 328-335, San Francisco, CA, 1995.

[SOA00] Simple Object Access Protocol (SOAP) 1.1. May 2000. Available at: http://www.w3.org/TR/SOAP

[Str99] Michael Strobel. Effects of electronic markets on negotiation processes - evaluating protocol suitability. Technical Report 93237, IBM, Zurich Research Laboratory, Switzerland, 1999.

[Str99b] Michael Strobel. On auctions as the negotiation paradigm of electronic markets. Technical Report 93212, IBM, Zurich Research Laboratory, Switzerland, 1999.

[Sur98] Gopalan Suresh Raj. A detailed comparison of CORBA, DCOM and Java/RMI, September 1998. Available at: http://www.execpc.com/gopalan/misc/compare.html.

[Tur97] Efraim Turban. Auctions and bidding on the Internet: an assessment. International Journal of Electronic Markets, 7(4), December 1997.

[UDD00] Universal Description, Discovery and Integration (UDDI). Technical white paper. September 2000. Available at: http://www.uddi.org.

[UML98] Rational Software Corporation. UML Documentation Set Version 1.1, Santa Clara, CA, 1998. Available at: http://www.rational.com/uml/.

[Var95] Hal R. Varian. Economic mechanism design for computerized agents. In USENIX Workshop on Electronic Commerce, pages 13-21, New York, NY, July 1995.

[Wfm02] WfMC: Workflow Management Coalition. http://www.wfmc.org.

[WfM99] Workflow Management Coalition, Terminology and Glossary. WFMC-TC-1011, February 1999, 3.0. Available at: http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf

[Wis98] Alexander Wise. Little-JIL 1.0 language report. Technical Report 024, Department of Computer Science, University of Massachusetts at Amherst, 1998.

[Wri97] Clive D. Wrigley, Design Criteria for electronic Markets Servers. Electronic Markets, 7(4):12-16, 1997.

[Wor98] The Workflow Automation Corporation. New Opportunities for Dramatic IT Results, a white paper. 1998. Available at: http://www.workflow.ca.

[WSD01] Web Services Description Language (WSDL) 1.1. March 2001. Available at: http://www.w3.org/TR/wsdl

[WSF01] Web Services Flow Language (WSFL) 1.0 by Frank Leymann. IBM Software Group. May 2001.

[WW98] Michael P. Wellman and Peter R. Wurman. Real time issues for Internet auctions. In IEEE Workshop on Dependable and Real-Time E-Commerce Systems, Denver, Co, June 1998.

[WWW98a] Peter R. Wurman, Michael P. Wellman, and William E. Walsh. The Michigan Internet AuctionBot: A configurable auction server for human and software agents. In 2nd International Conference on Autonomous Agents, pages 301-308, Minneapolis, MN, May 1998.

[WWW98b] Peter R. Wurman, William E. Walsh, and Michael P. Wellman. Flexible double auctions for electronic commerce: Theory and implementation. Decision Support Systems, 24(1):17–27, 1998.

[WWW+99] Peter R. Wurman, Michael P. Wellman, William E. Walsh, and Kevin A. O'Malley. Control architecture for a flexible Internet auction server. In IBM/IAC Workshop on Internet-Based Negotiation Technologies, Yorktown Heights, NY, March 1999.

[WZK00] Wai Yat Wong, Dong Mei Zhang, Mustapha Kara-Ali. Negotiating with Experience. In Proceedings of KBEM'00. Austin, Texas. July 2000.

[XMI98] XMI Metadata Interchange Specification. ftp://ftp.omg.org/pub/docs/ad/98-10-05.pdf.

[XML98] Extensible Markup Language (XML) 1.0 specification. http://www.w3.org/TR/1998/REC-xml-19980210.

[Yah02] Yahoo Auctions. http://www.auctions.yahoo.com.

[ZS98] Dajung Zeng and Katia Sycara. Bayesian learning in negotiation. International Journal of Human-Computer Studies, (48):125-141, 1998.