University Politehnica of Bucharest
Faculty of Automatic Control and Computers
Computer Science Department

Ph. D. THESIS

An Adaptive Negotiation Multi-Agent System for e-Commerce Applications

Ph. D. Student:
Șerban Radu

Scientific Advisor:
Prof. Adina Magda Florea

Bucharest, 2013

Contents

Figures Index

Tables Index

Chapter 1. Introduction

The main aim of this thesis is to conceive, design, implement, and evaluate different models of negotiation between autonomous agents, built to assist the users in automated negotiation for e-commerce.

The negotiation process is a complex feature of traditional buying and selling. This process can be examined in the context of automated negotiation, as applied in the multi-agent based e-commerce.

Creating and developing intelligent autonomous agents is an important issue nowadays. The agents could have different goals, constraints, capabilities and preferences. Negotiation between agents becomes a complex problem to be solved. The main approach for multi-agent bargaining is automated negotiation.

Because of the technology development, the business world encountered new possibilities to exchange data by computer networks at reduced costs. So, computer-based networking has created major changes in business activity. Therefore, these changes need a fundamental rethinking of the development of negotiation models. A business involves different processes, such as buying, selling, and services, which require the negotiation process. Software agents can be used in order to automate several of the most time consuming steps of the buying and selling processes.

The thesis realizes the modeling and design of a multi-agent system for automated negotiation, in which every agent has a cognitive part, composed from a knowledge base and an inference engine. The agents negotiate, based on a negotiation language, which contains a set of primitives, and also based on negotiation criteria, represented as rules.

The negotiation process is improved using learning algorithms. The use of learning techniques is investigated, in order to allow agents to reuse their negotiation experience for improving the final outcomes. The learning mechanism is used to improve the agent strategies in negotiation. Experimental results on four real world specific scenarios evaluate the performances of the multi-agent system for automated negotiation.

1.1. Problem Description

From the transactional point of view, there are the following types of electronic commerce:

a) **business-to-business** (**B2B**) - commerce transactions between businesses, such as between a manufacturer and a wholesaler, or between a wholesaler and a retailer. About 80 % of e-commerce is of this type;

b) **business-to-consumer** (**B2C**) – transaction that occurs between a company and a consumer;

c) **consumer-to-consumer** (**C2C**) – commerce between private individuals or consumers;

d) **consumer-to-business** (**C2B**) - consumers create value, and companies consume this value. Consumers can offer products and services to companies and the companies pay them;

e) **business-to-government (C2G)** – commerce between companies and the public sector;

f) **mobile commerce (m-commerce)** – buying and selling of goods and services through wireless technology.

Systems that use software agent technologies are proving to be effective in helping users make better decisions when buying or selling over the Internet. Software agents can also play an important role in providing automation and support for the negotiation stage of online commerce.

Agent-based systems support various stages of online commerce: product brokering (determining what to buy), merchant brokering (determining whom to buy from), and negotiation, where all parties involved communicate in order to reach an agreement on the terms of transactions. An example of such system is called Kasbah, proposed in [Chavez A. and Maes P., 1996]. More sophisticated automated trading systems are proposed, which offer multi-attribute intelligent matching, such as Tete-a-Tete [Guttman R. H. et al., 1998] and ITA [Kowalczyk R. and Bui V., 2000].

Bazaar is an experimental system for updating negotiation offers between two intelligent agents during bilateral negotiations [Zeng D. and Sycara K., 1998]. It models negotiation as a sequential decision making task, and uses bayesian probability as the underlying learning mechanism. The price is used as the issue of negotiation.

Most former studies on the negotiation are not for an automated negotiation system, but for a negotiation support system, that allows the negotiations between buyers and sellers. The reason is that in case of multi-issue negotiation, it is not easy to evaluate many negotiation issues, making the development of an automated negotiation system a complex problem.

Under the current e-commerce environment, an automated negotiation system is critical in dealing with complex problems and different changes in business environment [Choi H.R. et al., 2005]. The multi-agent frameworks have simple functions, such as the generation of agents, support for the conversation between agents and agents' management, but they don't have the function to support negotiation attributes.

Existing multi-agent frameworks have no function to define the category and relationship between products. These functions are necessary, if the conditions of a negotiation don't coincide. Negotiation messages should be prepared and used to bring a better result of the negotiation.

Negotiation can be viewed as a process of cooperative and competitive decision making between self-interested agents, in the presence of incomplete

information. It is a process in which competing agents decide how to divide the gains from cooperation [Fatima S. et al., 2007].

Machine learning is widely used in negotiation learning mechanisms. Negotiation parties use various learning techniques to improve and update their knowledge about environments and other parties, and the knowledge represents good support for their later decisions. The past negotiation records play an important role in all these learning techniques, because they compose the training data sets for the learning algorithms. As the larger training set implies more accurate learning results and increased profits, negotiation parties have the desire to obtain more related negotiation records. If several negotiation parties have the same negotiation learning objective and meanwhile they do not have any direct profit conflicts (for instance, a group of buyers conduct bilateral negotiations with a third party supplier), sharing their negotiation records will bring to all of them benefits [Lau R.Y.K., 2005]. Real-world negotiation scenarios, such as those found in B2B environments, are characterized by complex negotiation spaces, strict negotiation deadlines, limited information about the opponents, and different negotiator preferences. Therefore, practical negotiation systems must be equipped with effective learning mechanisms to acquire automatically domain knowledge, from the negotiation environments and continuously adapt to the dynamic negotiation contexts [Zhang S. and Makedon F., 2005].

Rule-based approaches represent a technique to parameterize the negotiation design space in multi-agent systems. Rules can be used for describing both strategies and mechanisms of automated negotiations [Badica C. et al., 2006 a].

1.2. Research Objectives

The main goal of the thesis is to develop a set of adaptive negotiation strategies, a model of autonomous entities that use these strategies and an associated implementation based on multi-agent system technology. The proposed model and implementation are aimed to support the development of advanced e-commerce applications, in which the users are represented by autonomous agents, which can adapt their negotiation strategies to both context and user preferences.

The framework is composed of a set of agents and a facilitator. The facilitator performs the function of exchanging messages and managing agents. The facilitator is in charge of a server function to exchange the messages between agents. The proposed strategies and models are validated by building an e-commerce environment that supports the development of several e-commerce scenarios.

When the proposed models are designed, the requirements for the negotiation protocol specified in [Bartolini C. et al., 2005] are followed, namely:

a) Be sufficiently formal that automated entities can interact using it;

b) Support negotiation about simple and complex objects;

c) Be sufficiently general that a variety of different market mechanisms can be expressed as specific instances of it;

d) Allow, but not require, the existence of a third party to arbitrate a given negotiation, for instance, a facilitator in an auction;

e) Support existing ways of doing business, as well as permitting more approaches in the future.

Regarding the negotiation strategies, some decision taking assistance techniques are used in the thesis, such as rule-based reasoning and machine learning techniques. The open environment allows automated negotiation, offering support for the use of various existing negotiation models, together with their respective negotiation strategies. In this manner, it is possible to integrate different models, approximating the automated negotiations to the way the real world interacts.

Multi-attribute techniques for negotiation are introduced by the agents, in the evaluation of the exchanged offers and counteroffers, and an associated communication mechanism is developed to support the negotiation protocol. Agents use multi-issue negotiation by exchanging offers and counteroffers, until they either reach a consensus that satisfies each party's private preferences and constraints, or the agents run out of offers and the negotiation fails. The agents use multi-attribute utility theory and constraint based reasoning for the evaluation and generation of offers. Multi-attribute negotiation is important for agents to reach agreements on multiple issues, but it is more complex than single-attribute negotiation.

The architecture for the negotiation environment is open and the number of buyers and sellers could be changed during execution time. In a flexible way, the negotiator agent could increase the number of negotiation strategies. These characteristics are designed in the environment using configuration files and rule-based systems. A flexible and interoperable system, where knowledge, negotiation protocol and strategies are explicitly represented in rules is developed.

The agents are rule-based and the rules express both the knowledge of the domain and the negotiation strategy. The rules are defined in *JESS* (Java Expert System Shell), for ensuring the interoperability among agents.

An agent is modeled as a knowledge-based system and is composed from the knowledge base, the inference engine and the control part. The knowledge base has two components, one component to represent the rules and knowledge about the objects of the business domain, and also one component to represent explicitly the strategy through negotiation rules.

In order to negotiate successfully, agents need to consider each others' agents preferences and generate offers accordingly. Agents may find each others' preferences over time and through interactions. As agents learn about each others' preferences, they can provide better offers and enable faster negotiation.

1.3. Outline of the Thesis

The thesis is composed of introduction, two parts with several chapters each of them, a chapter of conclusions and future work, bibliography and three annexes. The first chapter, ***Introduction***, contains the problem description, the research objectives, and this section.

The first part, ***Literature Overview on Automated Negotiation***, organized in three chapters, contains an overview of the most important theoretical concepts and technological aspects on which the thesis is based, and also presents the state of the art research in the relevant fields for the thesis contributions.

Chapter 2, ***Negotiation in Multi-Agent Systems***, describes the negotiation process in multi-agent systems and presents the possible negotiation types. Then, it provides guidelines for the models used in automated negotiation, the negotiation protocol and the negotiation strategies.

Chapter 3, ***Knowledge Representation and Learning***, contains an overview of the representation methods and languages that are mostly used in automated negotiation systems. Then, it describes the basis of some learning algorithms used in automated negotiation.

Chapter 4, ***Adaptive Negotiation Strategies***, is an overview of the current state of art in negotiation, with a focus on adaptive negotiation strategies. This provides a background, on which the personal contributions are built, and emphasizes significant results obtained and challenges for future research.

The second part, ***An Adaptive Negotiation Multi-Agent System***, contains the presentation of the contributions of the thesis for the design and implementation of an adaptive negotiation multi-agent system for e-commerce applications.

Chapter 5, ***Automated Negotiation using Profiles and Clustering of Agents***, defines an automated negotiation model, based on profiles and clustering of agents. In this model, the agents develop a set of negotiation profiles: the preference profile, the partner cooperation profile and the group-of-partners' negotiation profile. These profiles help the agents to conduct their negotiation. In the group-of-partners' negotiation profile, individual agent profiles are clustered, according to commonly discovered features.

Chapter 6, ***Automated Negotiation Model using Strategies and Tactics***, presents a model of heuristic negotiation between self-interested agents, which allows negotiation over multiple issues and learns the agent's negotiation strategy. The agents are using different strategies to negotiate and several models to adjust their decision during negotiation. The performance of the agents which use multiple tactics is compared to the agents having learning capabilities, based on reinforcement learning techniques.

Chapter 7, ***A Negotiation Model with BDI Agents***, describes a model of cognitive agents, based on the **BDI** (Belief-Desire-Intention) architecture, which takes into account different aspects of agent knowledge and behavior: abilities, history of

interaction with other agents, cooperation and negotiation options. This model defines the notion of utility for negotiation objects and for the roles the facilitator has in the negotiation process. The goal of the agents is to improve the negotiation strategy, using learning techniques and also to design negotiation environments.

Chapters 8 and 9 present different automated negotiation scenarios, corresponding to distinct business models, and their implementation in the system, showing the results obtained for each of them.

Chapter 8, ***Travel Agency Automated Negotiation Business Model***, describes a travel agency scenario, gives a detailed analysis of the negotiation rules and of the different strategies the agents may use, and reports experimental results for several use cases.

Chapter 9, ***Business Models Examples for Negotiation***, implements three other scenarios: a real estate agency scenario, a car dealer business model, and an emergency hospital negotiation scenario. For each scenario, the business case is described, and the experimental results obtained by implementing the scenario in the system are presented.

Chapter 10, ***Conclusions and Future Work***, presents the conclusions of the thesis, the original contributions of the author, and points out directions of future work.

The ***Bibliography*** contains 107 references, including publications of the author.

The ***Annexes 1, 2, and 3*** describe in details the set of negotiation rules for the scenarios defined in Chapter 9.

Part I. Literature Overwiev on Automated Negotiation

Chapter 2. Negotiation in Multi-Agent Systems

2.1. The Need for Negotiation in e-Commerce

Negotiation represents the process aimed to change the plans, in order to reach an agreement among a subset of businesses. In other words, negotiation is a form of decision making, where two or more agents search together a space of possible solutions through interaction, with the goal of reaching a consensus. Negotiation usually has a series of rounds, with every agent making a proposal at each round.

Negotiation is a process that appears in many aspects of our lives. Research in the field of automated negotiation has suggested the design and use of automated negotiators, on one hand to allow facilitation of the negotiation process by human negotiators and, on the other hand, to provide automated agents that can negotiate on behalf of humans [Lin R. et al., 2009 b].

Negotiation in electronic commerce is the process in which two or more parties multilaterally bargain goods or services for mutual intended gain, using the tools and techniques of electronic commerce [Beam C. and Segev A., 1997]. Automated negotiations take place when the negotiating function is performed by software agents.

A **software agent** is a computer program that acts for a user or other program. It represents an entity with goals, capable of actions endowed with domain knowledge and situated in an environment. **Multi-agent systems** are distributed agents that do not have the capabilities to achieve a goal alone and should communicate for this. They are suitable for the domains that involve interactions between different people or organizations with different goals.

In order to model automated negotiation, **negotiation protocols** and **negotiation strategies** should be differentiated. The **protocol** describes rules between negotiation participants, by specifying the requirements that enable their interaction. The **strategy** defines the behavior of participants, aiming to achieve a desired outcome. This behavior must be consistent with the negotiation protocol, and usually aims at maximizing individual gains of each of the negotiation agents [Badica C. et al., 2006 b].

A negotiation framework should specify a negotiation protocol, for constraining the use of the language. A protocol is a formal set of conventions governing the interaction among participants [Rahwan I. et al., 2003]. The negotiation protocol defines the formal interaction between the negotiators, whether the negotiation is done only once or repeatedly, and how the exchange of offers between the agents is conducted [Lin R. and Kraus S., 2010]. The negotiation strategy is an important issue. If one agent's negotiation strategy is known to the other agent, the first agent

may be in a significant disadvantage. Strategy means an analysis of the counterpart's negotiation strategy and an offer for a responding proposal.

There are three major approaches to automated negotiation in the multi-agent domain: **game-theoretic** approach, **heuristic-based** approach, and **argumentation-based** approach.

Game-theoretic approaches of automated negotiation assume that agents have unbounded computational resources and that the space of outcomes is completely known. In most real environments, these assumptions fail, due to the limited processing and communication capabilities of the information systems.

Heuristic-based approaches overcome the shortcomings of game-theoretic approaches, but they have a number of disadvantages. The models lead to outcomes that are sub-optimal, because they adopt an approximate notion of rationality and because they do not examine the full space of possible outcomes. Also, it is very difficult to predict how the system and the agents will behave. The heuristic negotiation strategies are domain dependent and time consuming. A reinforcement learning approach allows the negotiator to learn which negotiation primitive to use in a certain state of the negotiation [Florea A.M. and Kalisz E., 2008].

Argumentation-based approaches attempt to overcome the limitations of other approaches, by allowing agents to exchange additional information, or to argue about their beliefs and other mental attitudes during the negotiation process. In the context of negotiation, an argument is a piece of information that may allow an agent to justify its negotiation position or to influence another agent's negotiation position [Rahwan I. et al., 2003].

The agents have limited information about the preferences and constraints of each other. They make decisions according to available information about private preferences, constraints and individual negotiation strategies. The agents exchange information in the form of offers. An offer is a complete solution, which is currently preferred by an agent, given its preferences, constraints and the negotiation history of offers and counteroffers.

An agreement takes place when a particular offer is accepted by all negotiation parties. During the negotiation process, the range of possible offers of each agent changes, according to the current information available. These give finally an agreement, or, if a deal is not possible, the negotiation ends unsuccessfully. Therefore, negotiation is typically an iterative process of evaluating the offers, updating the available options, and making the counteroffers, according to the individual negotiation strategies.

An automated negotiation system has three steps: the estimation step, the negotiation making step, and the negotiation concluding step. In the stage of an estimate, upon the buyer's request for a written estimate, the seller reviews his process planning, performing cost accounting, and suggests his estimate to the buyer. During the stage of making negotiations and concluding negotiations, one

agent evaluates the other agent's negotiation proposal, building one's own negotiation strategy and sending his proposal to the other.

The cooperation between agents should be made by means of a common message through which the request for a deal will be able to be made and the result to be sent. In general, messages between agents under the multi-agent environment are based on **FIPA-ACL** (Foundation for Intelligent Physical Agents - Agent Communication Language) and **KQML** (Knowledge Query and Manipulation Language). FIPA-ACL is a standard language for agent communications. KQML is a language and protocol for communication among software agents and knowledge-based systems.

Effective and efficient multi-issue negotiation requires an agent to have some indication of its opponent's preferences. However, in competitive domains, such as e-commerce, an agent will not reveal this information and so the best that can be achieved is to learn some approximation of it through the negotiation exchanges [Coehoorn R. and Jennings N., 2004].

For instance, if it is considered an organization that needs to order some products, it could have a buying agent that moves through all the stages of the buying process. An agent can be designed to automatically gather information on sellers and products, which may match the requirements of that organization. After the evaluation of various offers, a decision is made about which sellers and products to investigate. This is followed by negotiation on the terms of the transactions with these sellers. Following successful negotiation, orders are placed and payment is made automatically.

The negotiation process appears in all electronic transactions at the time of the communication of agents in order to reach mutual beneficial agreements. The agents might have some common interest in cooperating, but might have some conflicting issues about how to cooperate, as shown in Figure 2.1 [Florea A.M., 2012]. Agents can mutually benefit in reaching agreement on a particular result from a set of possible outcomes, but might have some conflicting interests to overcome in achieving the result that they prefer. Before moving into any cooperation, they need to decide how to cooperate in order to obtain the associated benefits. On the other hand, each agent would like to reach an agreement that is favorable to itself as quickly as possible. Therefore, they need to make a series of offers and counteroffers before any agreement is reached.

While software agents not only save the time of human negotiators, they also find solutions that are as beneficial as possible to all the parties. As a result, negotiation in an electronic environment increases the efficiency of negotiations through the assistance and automation of decision tasks.

This increase in efficiency provides the following benefits [Huq G., 2010]:

a) The complexity and uncertainty of non-automated decisions in electronic negotiation can be reduced due to adequate information being provided;

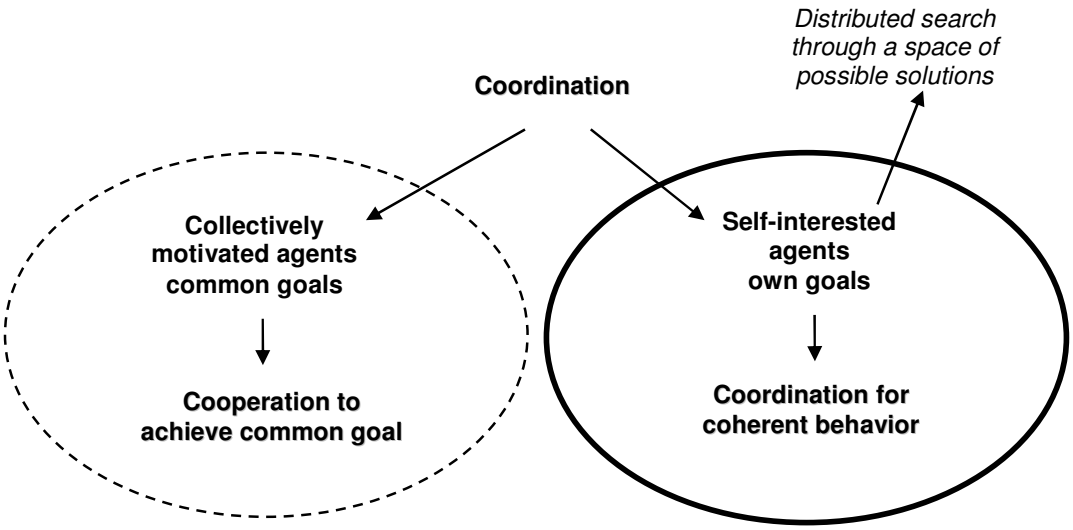


Figure 2.1. Coordination in Negotiation [Florea A.M., 2012]

b) Structured negotiation tasks with a well-defined solution approach can be applied;

c) Increase the total number of potential transaction participants, which offers more options, flexibility and eventually attract more efficient agreements, when moving towards negotiation;

d) More transparency can be enabled between participants;

e) Reduce the total cost and time of negotiation.

Possible scenarios for automated negotiation [Rahwan I. et al., 2002 b] are presented in the next paragraphs.

In a telecommunications market, a software agent, representing the user and running on a smart phone, negotiates with other software agents representing phone companies. This is presented in Figure 2.2 (a). The user agent attempts to find a good quality phone connection for a reasonable price. Each phone company agent aims at maximising its company's profit. Conflict of interests exists because the user competes on money and bandwidth with various service providers, while service providers themselves compete over user's money.

In dynamic supply chains, a software agent acting on behalf of a computer manufacturer negotiates with various supplier agents, in order to assure the deliver of various components. Each supplier agent might itself negotiate with subcontractors to get the components it needs. This is presented in Figure 2.2 (b).

In this scenario, the negotiation mechanism involves allocating money and computer components. Each part aims at making more money and the different monitor and printer cartridge suppliers compete over contracts with the computer and printer manufacturers respectively.

Agents may begin to negotiate without having complete and accurate preferences over alternative deals. This may be because agents have limited,

uncertain, or wrong information, due to imperfect sensing of the environment or because they lack the necessary time. As a result, agents may have incomplete or incorrect preferences over different deals. Negotiating on the basis of such preferences can lead to sub-optimal deals.

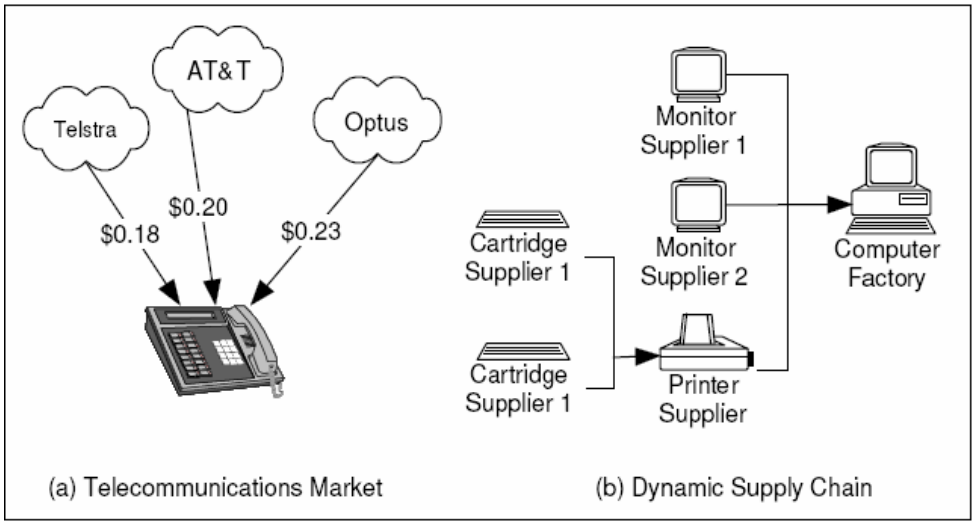


Figure 2.2. Automated Negotiation Scenarios [Rahwan I. et al., 2002 b]

For example, an agent organizing a trip from Bucharest to Wien on behalf of a user, as shown in Figure 2.3, might not know about all possible ways of travelling to Wien. Further, the user might have forgotten to request a hotel booking. Due to such incomplete and inaccurate information, the agent can make incorrect decisions as it negotiates with different travel agents.

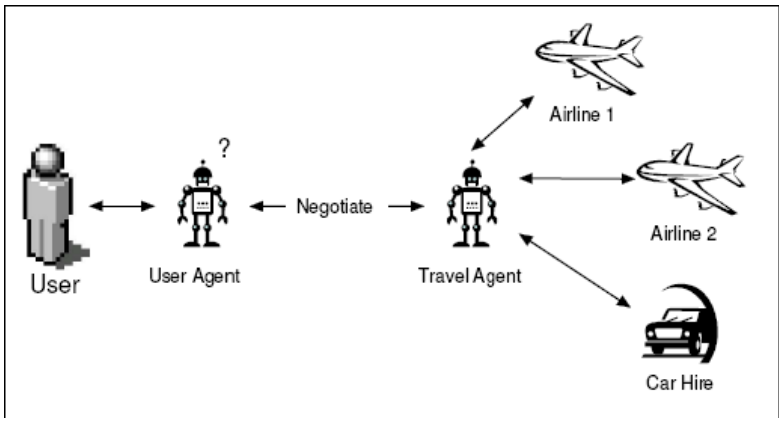


Figure 2.3. Travel Agent Scenario [Rahwan I. et al., 2002 b]

2.2. Modeling the Negotiation in e-Commerce

Electronic commerce is one of the most important market places in today's electronic environment, where buyers and sellers are involved in trading activities. Electronic commerce is expanding and becoming more popular to both business organizations and consumers.

A multi-agent system has a set of agents, which have some interactions between them, in general by messages exchange. Agents in a multi-agent system represent or act on behalf of users or owners with a diversity of goals and motivations. Therefore, these agents require the ability to cooperate, coordinate and negotiate with each other for successful interaction.

Negotiation is a process by which a group of agents communicate to try to come to a mutually acceptable agreement on some matter. It is one of important methods for establishing agent cooperation.

Negotiation refers to the process by which a group of agents communicate with one another in order to reach a mutually acceptable agreement. If the negotiation spaces are large, even experienced human negotiators are overwhelmed. Under such circumstance, sub-optimal rather than optimal deals are often reached. Therefore, it is desirable to employ intelligent software agents to automatically search through the negotiation spaces to find potential agreements on behalf of the human negotiators.

Negotiation between agents appears in different areas of research, like electronic commerce, distributed resource allocation or virtual enterprises. In open systems, agents are acting in an environment in which other agents may enter or leave, some of them known before, some others encountered for the first time. In this context, the design of intelligent agents with a complete pre-defined negotiating behavior represents a challenge for the designer, especially when the agents are conceived to be general purpose and not limited to a specified domain. To overcome existing difficulties, creating automatic negotiating agents is still a fertile area of research, despite the important amount of work in the domain.

The environments of applications are open, as they are populated with self-interested agents designed and/or owned by different people and there is no complete information about the preferences or decision-making processes of the participating agents. In order to be autonomous and achieve performance when conducting a negotiation, an agent should be able to anticipate both the outcome of the negotiation and the best potential partner with which to start a negotiation. Machine learning approaches can contribute to adapt the agent's strategy during negotiation and trading, achieve improved outcomes and increased payoffs.

Negotiation plays an important role in multi-agent systems. When one business organization wants to buy or sell goods or have services in an electronic environment [Huq G., 2010], then it always needs some processes that involve negotiation, as presented in Figure 2.4.

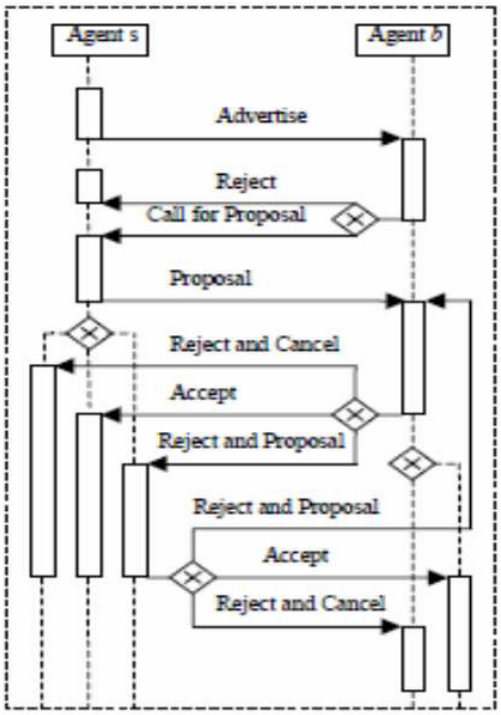


Figure 2.4. Negotiation Steps between Two Agents [Huq G., 2010]

The components, which form the negotiation setting, are [Wooldridge M., 2009]:

a) the **negotiation set**, representing the space of all possible proposals that agent can make;

b) the **negotiation protocol**, describing the proposals that agents can make, with respect to the previous negotiation history;

c) the **negotiation strategy**, determining what proposals the agents will make. In general, each agent has its own strategy. The fact that an agent is using a certain strategy is in general not visible to other negotiation participants;

d) the **rule** which determines when a deal is obtained and what the agreement deal is.

In general, negotiation has a series of rounds, with some proposal made at every round. The proposals that agents make are defined by their strategy, should be derived from the negotiation set, and should be correct, as defined by the protocol. If agreement is obtained, as described by the agreement rule, then negotiation ends with the agreement deal.

The negotiation complexity increases, as the number of agents involved in negotiation increases. There are three negotiation types:

a) **one-to-one** negotiation, in which one agent negotiates with another agent. A simple case of one-to-one negotiation is when the agents have symmetric preferences with respect to the possible deals;

b) **many-to-one** negotiation, in which many agents negotiate with one agent. Auctions are an example of many-to-one negotiation. Such a negotiation can be seen as a many one-to-one negotiations done in parallel;

c) **many-to-many** negotiation, in which many agents negotiate with other agents simultaneously. If there are **n** agents involved in negotiation, it means that there can be up to **n*(n-1)/2** negotiation threads.

Also, the negotiation complexity increases in the case of **multi-issue** negotiations. An instance of a **single-issue** negotiation is when two agents are negotiating only the price of a certain good or service. In such a scenario, the preferences of the agents are symmetric, such that a deal, which is more preferred from one agent's point of view, is guaranteed to be less preferred from the other agent's point of view. Such a symmetric scenario is simple to be analyzed, because it is clear what represents a concession. For the seller to concede, it must decrease the price of its proposal, while for the buyer to concede, it must increase the price of its proposal. In **multi-issue** negotiation scenarios, agents negotiate not only the value of a single attribute, such as price, but the values of multiple attributes, which may be interrelated. In multi-issue negotiations, it is not clear what represents a true concession and when all the attribute values must be either increased or decreased. Multiple attributes give an exponential growth in the space of possible deals.

There are **five characteristics** that are necessary for a negotiation mechanism: efficiency, stability, simplicity, distributivity and symmetry [Benameur H. et al., 2002]. These characteristics are found when considering a model for automatic negotiation:

a) **Efficiency** - expressing the efficiency of a negotiation is a difficult task, because there must be taken into account restrictions;

b) **Stability** - a possibility to get stability is to not allow an agent to ignore or reconsider offers, once they have been submitted;

c) **Simplicity** - negotiation mechanisms should be simple to implement. Taking into account the set of messages to be exchanged, the communications needed are the offers and the answers from both parts;

d) **Distributivity** - when many buyers are active, the seller is the main entity. But in a market, a lot of buyers and sellers could negotiate. The sellers could act independently or coordinate their activities. Multiple simultaneous negotiations can be performed in this way;

e) **Symmetry** - the symmetry could be obtained by assuming that all agents could access all the available information.

For conducting business efficiently, there is an interaction from one business organization to other business organization. The suppliers, manufacturers, retailers and consumers, are all in a related network, which needs proper, efficient and timely coordination, cooperation and negotiation processes. Therefore, in an electronic environment, when the above entities interact with each other, the system needs different automated software agents to perform tasks on behalf of real-world

business organizations. This can be achieved by applying multi-agent systems in an electronic environment, in order to improve the performance among software entities.

New strategies are required for proper planning and managing the businesses. In order to develop these strategies, automation of business processes is required. The negotiation process is one of the most important processes in business communities. In addition, automation of negotiation, which corresponds to negotiation-based e-commerce, has received a lot of attention from the multi-agent community, because such topics have the important potential to reduce significantly the negotiation time and to remove some of the reticence of humans to engage in negotiation and to facilitate the intelligent agents that are able to perform negotiation on behalf of users.

In market environments, business activities are coordinated through price, which is one of the values that is assigned to a negotiation object. Various businesses assign different values to negotiation objects, and for this reason, they need to bargain to reach mutually acceptable agreements. Therefore, it is important to enable businesses to engage in negotiation in an electronic business environment. There are many areas of research that studied negotiation in an electronic environment, like: the information systems field with negotiation support systems; the multi-agent systems field with searching, trading and negotiating agents, and the market design field with electronic auctions [Zlatev Z. et al., 2004].

The negotiation process can be decomposed into components, such as negotiation protocols, negotiation objects and agent's decision-making models. Negotiation protocols describe the rules used in negotiation. Negotiation objects refer to the issues the agents are negotiating, that is, goods or services. Agent's decision-making models represent decision-making framework, which agents use to fulfill their goals with respect to the protocol.

Real negotiation situations are characterized by complex negotiation spaces, which involve multiple parties and many issues. In addition, negotiators are bounded by limited computational resources, time, and limited information about the opponents. While classical game-theoretic negotiation models provide good theoretical analysis of the optimal outcomes, these fail to advise the course of actions that a negotiator can follow to reach the optimal outcome in real world negotiations. One main concern for the practical use of these theories is that the search space for considering all the possible strategies and interactions, in order to identify the equilibrium solutions grows exponentially. It means that the problem of finding an optimal strategy is in general computationally difficult. Another problem is that the classical game theories assume that complete information about every agent is available to a centralized decision making mechanism. It turns out that such an assumption does not hold in most real-world negotiation situations.

2.3. The Coordination Process in Negotiation

As the number of electronic transactions increases, the interest for partial or full automation of these transactions also grows. E-commerce describes the changes that are transforming the way business is conducted, through the use of information technology. In order to understand the basic stages of an e-commerce procedure, it is examined a **consumer's buying behavior model** [Guttman R.H. et al., 1998; Ye Y. et al., 2001].

In such a model, there are **six steps**, which appear during the buying process. **Need identification** is done by the buyer, so he needs a particular product. **Product brokering** is the information gathering by the buyer, so that he is able to decide what to buy. The buyer provides some criteria and a filtering mechanism is employed to provide him with a set of products. **Merchant brokering** combines the products set from the previous stage with information for specific merchants and gives the buyer a set of potential sellers. **Negotiation** is the stage during which both participants try to reach an agreement, over possible negotiation issues, such as the price, the volume or the delivery time of a product. **Purchase and delivery** follow the negotiation phase and payment or delivery options are examined. Finally, **product service and evaluation** is the evaluation of product and customer service, made by the buyer in order to estimate his utility.

The main techniques used for product brokerage are: **characteristics filtering**, **collaborative filtering** and **constrained-based filtering** [Guttman R.H. and Maes P., 1998]. Characteristics filtering select the products based on associated keywords or characteristics. Collaborative filtering refers at sending personalized recommendations to an agent, based on similarities between different profiles of users' preferences. Constrained-based filtering implies an agent, which specifies the imposed constraints upon the product.

In the current negotiation environments, which represent the first generation of e-commerce applications, buyers and sellers are humans who browse through a catalogue of well-defined goods and make fixed price purchases, usually by means of credit card transaction [He M. et al., 2003]. Humans appear in all the stages of the buying process, and this is time consuming. The research is conducted towards the realization of the second generation, which is the future state, of e-commerce applications, which will be done through the use of automated methods of information technology. Web users will be represented by software agents. Nowadays, there is an increasing use of software agents for all the aspects of e-commerce.

However, as software agents start to engage in e-commerce, new issues arise. Information must be organized in a way that is accessible by both humans and machines. Additionally, machines must be able to access, process and interpret the information in the same way. This vision is consistent with the **Semantic Web** initiative, which enriches the current Web through the use of machine-processing information about the semantics of information content. This way, the meaning of

displayed information is accessible not only to humans, but also becomes accessible to software agents. The techniques of the Semantic Web are semantic annotations (meta-data) and ontologies, which organize terms in a conceptualization of a domain, thus connecting semantic annotations with each other and serving as a basis for interoperability.

Electronic commerce is rapidly gaining acceptance nowadays, and it will become more widespread in the future. Current e-commerce systems, such as **Amazon.com**, allow a user to browse an online catalog of products, choose some, and then purchase these selected products using a credit card. However, agents can lead to the second-generation of e-commerce systems, where many aspects of consumer buying behaviors (in B2C systems) and business-to-business transactions (in B2B systems) are automated. The automation on both the buyer's and the seller's side can lead to applications that are more dynamic and personalized. Both buyers and sellers gain from these changes - the buyers can expect the agents to search for and retrieve the best deals available, while the sellers can have the agents that automatically customize their offerings to customers, based on various parameters, such as the customer type, current seller competition, and current state of the seller's own business. This advanced degree of automation can be achieved by modeling the e-commerce systems as interacting agents.

The agents should go through a coordination process involving negotiations over all possible agreements covering issues of common interest, eventually bringing them all to a consensus. There are three main issues in defining such a coordination process [Goradia H.J. and Vidal J.M., 2007]:

a) **Space of Possible Deals** - this represents a finite set of candidate deals for the agents to consider. Possible proposals that the agents can make are restricted by this set;

b) **Negotiation Process** - this is a negotiation protocol, which, given the set of possible deals, defines how the agents will find to an agreement on a single deal. It specifies the set of rules that govern the agent interactions, while they attempt to reach a consensus. The process explicitly defines the various negotiation states, the events that cause negotiation states to change, and the valid actions for the agents in particular states. The negotiation process also defines the rules that determine when a deal is obtained, and what this agreement deal is;

c) **Negotiation Strategy** - given a set of possible deals and a negotiation process, a negotiation strategy represents a model that individual agents employ to make decisions and achieve their objectives. The negotiation protocol, as well as certain agent characteristics, whether the agent has complete knowledge of its environment, whether it is truthful, determines the complexity of the agent decision model.

The above parameters lead to a rich and complex environment for analyzing negotiation problems. Negotiations typically involve a series of rounds, with each agent making a proposal at every round, before eventually converging to an

agreement. However, there are many attributes that can complicate this model in real-world negotiation settings. For example, the complexity of the model increases rapidly with the number of agents in the negotiation process. Not only the negotiation space increases, but also the negotiation strategies of the individual agents become more complex. Also, the agents in real-world settings typically have multiple issues over which agreement must be reached. Suppose there are **n** issues that the agents have to negotiate, where each issue can have **m** possible values. This leads to a set of **m*n** possible deals. To make matters worse, in many situations the number of issues in the negotiation process may itself vary based on other parameters.

2.4. Types of Negotiation

2.4.1. Negotiation based on Game Theory

Game theory studies interactions between self-interested agents [Jennings N.R. et al., 2001]. Game theory is important in the automated negotiation research, because the agents in these negotiations are self-interested. In order for an agent to make the choice that optimizes its outcome, it must reason strategically. That is, it must take into account the decisions that other agents may make, and must assume that they will act so as to optimize their own outcome. In negotiation, this means, for example, taking into account the private valuations that agents have on the negotiation issues, their own deadlines for making a deal, and so on. Game theory gives a way of formalizing and analyzing such concerns.

Game theoretic techniques can be applied to two key problems:

a) The design of an appropriate protocol that conducts the interactions between the negotiation participants;

b) The design of a particular strategy that individual agents can use while negotiating – an agent aims to use a strategy that maximizes its own individual welfare.

There are a number of problems associated with the use of game theory when applied to automated negotiation [Jennings N.R. et al., 2000]:

a) game theory assumes that it is possible to characterize an agent' preferences with respect to possible outcomes. Humans, however, find difficult to define consistently their preferences over outcomes. In general, human preferences cannot be characterized even by a simple ordering over outcomes. In scenarios where preferences are obvious, game theoretic techniques may work well. With more complex preferences, it is much harder to use them;

b) the theory has failed to generate a general model governing rational choice in interdependent situations;

c) game theory models often assume perfect computational rationality, meaning that no computation is required to find mutually acceptable solutions within a feasible range of outcomes. Furthermore, this space of possible deals is often assumed to be

fully known by the agents, as the potential outcome values are. This assumption is seldom true in most real world cases. Agents know their own information space, but they do not know that of their opponent.

In the negotiation based on game theory, criteria to evaluate negotiation protocols among self-interested agents are used. The agents are supposed to behave rationally, meaning that an agent prefers a greater utility or payoff, over a smaller one. When referring to payoff maximization, it is possible to have individual payoffs, group payoffs or social welfare.

Social welfare deals with the sum of agents' utilities or payoffs in a given solution. It measures the global well-being of the agents. The problem arises when comparing the utilities.

A solution **x**, that is a payoff vector $\mathbf{p(x_1, \ldots, x_n)}$ is called **Pareto efficient** or **Pareto optimal**, if there is no other solution **x'** such that at least one agent is better off in **x'** than in **x** and no agent is worst off in **x'** than in **x**. It measures the global well-being, it does not require utility.

The **individual rationality** (**IR**) of an agent' participation refers to the agent's payoff in the negotiated solution, which should be no less than the payoff that the agent would get by not participating in the negotiation. A mechanism is IR if the participation is IR for all the agents.

A protocol is called **stable** if once the agents arrived at a solution, they do not deviate from it. In a **dominant strategy**, an agent is best off using a specific strategy, no matter what strategies the other agents use.

Suppose that $\mathbf{r = f(Act_A, Act_B)}$ is the result (state) of actions $\mathbf{Act_A}$ of agent **A** and $\mathbf{Act_B}$ of agent **B**. A strategy $\mathbf{S_1 = \{r_{11}, r_{12}, \ldots, r_{1n}\}}$ **dominates** another strategy $\mathbf{S_2 = \{r_{21}, r_{22}, \ldots, r_{2m}\}}$ if any result of $\mathbf{r \in S_1}$ is preferred (better than) to any result of $\mathbf{r' \in S_2}$.

Two strategies, $\mathbf{S_1}$ of agent **A** and $\mathbf{S_2}$ of agent **B**, are in a **Nash equilibrium** if:

a) in case agent **A** follows $\mathbf{S_1}$, agent **B** can not do better than using $\mathbf{S_2}$ and

b) in case agent **B** follows $\mathbf{S_2}$, agent **A** can not do better than using $\mathbf{S_1}$.

The previous definition can be generalized for several agents using strategies $\mathbf{S_1, S_2, \ldots, S_k}$. The set of strategies $\mathbf{\{S_1, S_2, \ldots, S_k\}}$ used by the agents $\mathbf{A_1, A_2, \ldots, A_k}$ is in a **Nash equilibrium** if, for any agent $\mathbf{A_i}$, the strategy $\mathbf{S_i}$ is the best strategy to be followed by $\mathbf{A_i}$, if the other agents are using strategies $\mathbf{\{S_1, S_2, \ldots, S_{i-1}, S_{i+1}, \ldots, S_k\}}$.

A **mixed strategy** $\mathbf{p_i}$ of a player **i** is a probability distribution over actions $\mathbf{A_i}$ available to **I**. A **pure Nash equilibrium** is a Nash equilibrium using pure strategies. A **mixed Nash equilibrium** is a Nash equilibrium using mixed strategies. A **mixed Nash equilibrium** is a set of mixed strategies, one for each agent, so that no agent has an incentive to unilaterally deviate from its assigned strategy.

In a transaction, when a buyer and a seller agent value a product differently, a surplus is created. A bargaining solution is a way in which buyers and sellers agree to divide the surplus. Trade would result in the generation of surplus, while no surplus is created in case of no trade. A bargaining solution provides an acceptable way to

divide the surplus among the two parties. The bargaining problem is described in Figure 2.5.

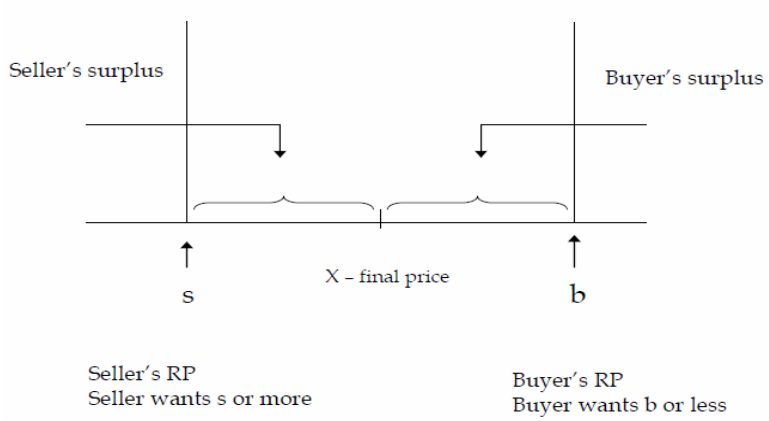


Figure 2.5. Bargaining between Buyer and Seller Agents [Florea A.M., 2012]

A **bargaining solution** is defined by $\mathbf{F:(X,d)\rightarrow S}$, where $\mathbf{X\subseteq\Re^2}$ and $\mathbf{S,d\subseteq\Re^2}$. **X** represents the utilities of the players in the set of possible bargaining agreements and **d** represents the **point of disagreement**. For instance, if the price for a certain item is between **10** and **20** monetary units, then the bargaining set is $\mathbf{x+y \leq 10}$, $\mathbf{x \geq 0}$, $\mathbf{y \geq 0}$. A point **(x,y)** in the bargaining set represents the case when the seller gets a surplus of **x**, and the buyer gets a surplus of **y**, that is the seller sells the item at **10+x**, while the buyer pays **20-y**.

2.4.2. Heuristic based Negotiation

A way of overcoming the limitations of game theoretic models is to use heuristic methods. Such models acknowledge that there is a cost associated with computation and decision making and seek to search the negotiation space in a non-exhaustive manner. This has the effect that heuristic methods aim to produce good, rather than optimal solutions. The methods themselves may either be computational approximations of game theoretic techniques or they may be computational realizations of more informal negotiation models. In heuristic negotiation, there is no central mediator and the messages are private between the negotiating agents. In general, the protocol does not prescribe an optimal course of action. The main concern is represented by the agent's decision making heuristic model during the course of negotiation.

In the heuristic approach, the models are based on realistic assumptions and they provide a more suitable basis for automation and they can, therefore, be used in a wider variety of application domains [Jennings N.R. et al., 2001].

The space of possible agreements is quantitatively represented by contracts, having different values for each issue. Every agent evaluates these points in the space of possible outcomes, according to some preference structure, captured by a utility function. Proposals and counterproposals are offers over single points in this space of possible results. Search terminates either when the time to reach an agreement has been exceeded or when a mutually acceptable solution has been reached.

In heuristic negotiation, the models often select outcomes that are sub-optimal, because they adopt an approximate notion of rationality and because they do not examine the full space of possible results. Also, the models need extensive evaluation, through simulations and empirical analysis, since it is almost impossible to predict precisely how the system and the agents behave in a wide variety of circumstances.

In heuristic negotiation, a **negotiation object** (**NO**) can be defined as the range of issues over which agreements must be reached. For instance, the object of negotiation may be an action which the negotiator agent *A* asks another agent *B* to perform for it, a service that agent *A* asks to *B*, or an offer of a service agent *A* is willing to perform for *B*, provided that *B* agrees to the conditions of *A* [Florea A.M., 2012]. The negotiation primitives used in such a scenario can be the following:

a) **Request NO** – request of a negotiation object;

b) **Accept name(NO)** – accept the request for the **NO**;

c) **Reject name(NO)** – reject the request for the **NO**;

d) **ModReq name(NO) value(NO, X, V_1)** – modify the request by modifying the value of the attribute **X** of the **NO** to a different value **V_1**.

The protocol used by the agents for the primitives defined above is presented in the Figure 2.6.

2.4.3. Argumentation based Negotiation

This approach allows additional information to be exchanged. This information is of different forms, mainly arguments which explain the opinion of the agent. Thus, in addition to reject a proposal, an agent can offer a critique of the proposal, explaining why it is unacceptable.

When evaluating an argument, the agent needs to assess the argument on its own merits and then modify this by its own perception of the argument's degree of credibility, in order to work out how to respond [Mercier H. and Sperber D., 2011].

Using argumentation means handling the complexities of the agents' mental attitudes, the communication between agents, and the integration of the argumentation mechanisms into a complex agent architecture.

Suitable argumentation protocols should be defined, that is, set of rules that specify how agents generate and respond to arguments, based upon what they know.

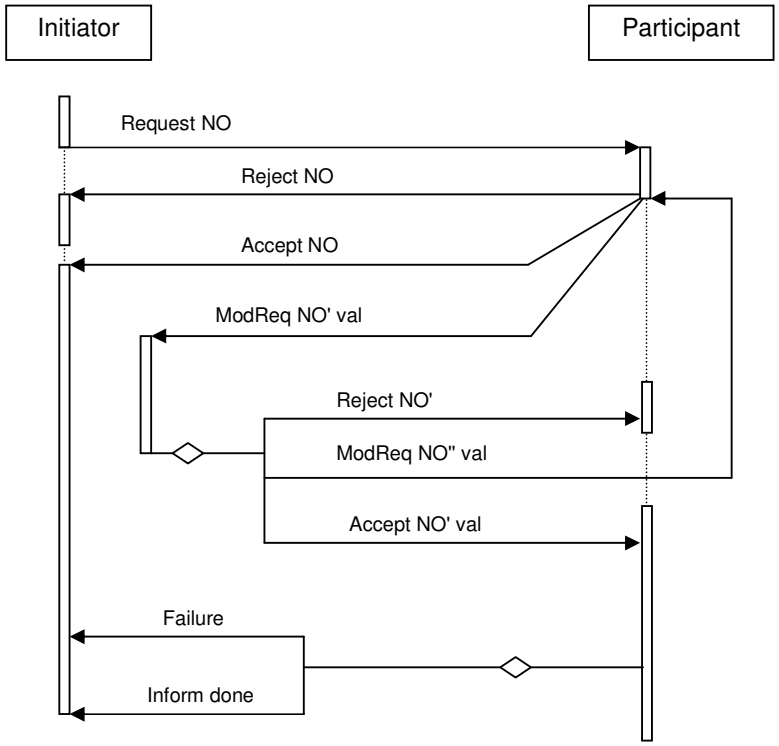


Figure 2.6. The Protocol for the Defined Primitives [Florea A.M., 2012]

Argumentation based negotiator agents have the ability to be persuasive and so achieve agreements, which non-argumentation based negotiators can't reach. However, the problem is the overhead added to the negotiation process.

The arguments are used to persuade the opponent to accept a negotiation proposal. There are different types of arguments. Each argument type defines preconditions for its usage. If the preconditions are met, then the agent may use the argument. The agents need a strategy to decide which argument to use. The following arguments can be used by the agents, presented together with the associated negotiation primitive [Florea A.M. and Kalisz E., 2008]:

a) **Appeal to past promise** - the agent **A** reminds agent **B** of a past promise regarding the negotiation object (**NO**), that is, agent **B** has promised to the agent **A** to perform or offer the **NO** in a previous negotiation. *Preconditions:* **A** must check if a promise of the **NO** (future reward) was received in the past in a successfully concluded negotiation. *Negotiation primitive:* **Remember NO**;

b) **Promise of a future reward** - the agent **A** promises to do a **NO** for the other agent **B** at a future time. *Preconditions:* **A** must find one desire of agent **B** for a future time interval, if possible a desire, which can be satisfied through an action that **A** can perform, while **B** can not. *Negotiation primitive:* **Promise NO**;

c) **Appeal to self interest** - the agent **A** believes that concluding the contract for **NO** is in the best interest of **B** and tries to persuade **B** of this fact. *Preconditions:* **A** must find one of **B** desires, which is satisfied if **B** has the **NO** or, alternatively, **A**

must find another negotiation object **NO'** that is previously offered on the market and it believes **NO** is better than **NO'**. *Negotiation primitive:* **CompareD NO Desire** or **CompareO NO NO'**;

d) **Threat** - the agent **A** makes the threat of refusing doing/offering something to **B** or threatens that it will do something to contradict **B**'s desires. *Preconditions:* **A** must find one of **B**'s desires directly fulfilled by a **NO** that **A** can offer or **A** must find an action that is contradictory to what it believes is one of **B**'s desires. *Negotiation primitive:* **TreatForbid NO** or **ThreatDo NO**.

2.5. Modeling the Negotiation in Multi-Agent Systems

The agents should be designed and implemented such that to negotiate concurrently with other agents. Negotiating agents are developed to improve the set of possible agreements and the concession amount they are willing to do, like an answer to different situations and negotiation conditions. In this perspective, agents have a time-dependent negotiation strategy, in which the private value of each negotiation object is dynamically found by: the probability that the negotiation will not be successfully concluded, the expected agreement price of the negotiation object, and the expected number of deals [Silva S., 1996].

Agents' negotiation strategies are performed in dynamic and complex negotiation environments, in which agents have conflicting objectives and preferences. Also, they have incomplete information about other agents and have multiple trading partners and trading competitors [An B. et al., 2010].

From a theoretical perspective, agents' strategies in negotiation are analyzed. Game theoretic analysis provides insights and theoretical foundations for developing negotiation agents. For real complex dynamic negotiation involving multi-agents, it is impractical to compute agents' rational strategies and heuristic based negotiation strategies are designed.

2.5.1. Automated Negotiation Modeling

In the negotiation theory, most work focuses on bilateral negotiation [Tamma V. et al., 2005; An B., et al., 2009; An B. et al., 2010]. One-to-many and many-to-many negotiations are also important and widely exist in many application domains. For one-to-many negotiation, an auction is widely used and, for many-to-many negotiation, market mechanisms like matching or two-sided auction are appropriate. Even if an agent interacts with many agents, a common assumption is that an agent can be involved in only one negotiation at a time. The result is that an agent may finish the current negotiation in disagreement, in spite of possible gains from bargaining, in order to find a more attractive alternative. Therefore, the idea that an agent is involved in only one negotiation at a time appears to be restrictive.

Agents' strategic behavior in one-to-many and many-to-many negotiations are analyzed, in which agents are negotiating with multiple trading partners and, at the same time, are facing competition from trading competitors.

The analysis shows that both negotiation order and market competition affect agents' negotiation power. An agent's negotiation power increases with the number of trading partners and decreases with the number of trading competitors.

In e-commerce environments, in which self-interested agents act individually, agents should obtain different goods or services. Therefore, agents may need to engage in multiple negotiations. If these negotiations are not all successful, users gain decrease. From the perspective of the overall negotiation, goods or services are dependent, as an agent's utility from the overall negotiation depends on obtaining overall agreements on all the transactions. The negotiation environment has the following three features [An B. et al., 2010]:

a) When acquiring multiple goods, a buyer agent only knows the private value available for the set of items, that is the highest price the agent can pay for all the goods, rather than the private value of each separate item;

b) Agents can decommit from tentative agreements at the cost of paying a penalty. Decommitment allows agents to profitably accommodate new negotiations. If these negotiations make some existing contracts less profitable or infeasible for an agent, that agent can decommit from those contracts;

c) Negotiation agents are assumed to have incomplete information about other agents, for example, a buyer agent knows the distribution of the private value of a seller agent and the number of trading competitors. However, an agent's negotiation status (the set of proposals it has received) and negotiation strategy are its private information. During negotiation, the agents can quit negotiation at any time, even without notifying their trading partners. When an agent wants to buy multiple goods or services, it concurrently negotiates with sellers to reach agreements for all the items.

In order to evaluate the performance of negotiation agents, simulation environments consisting of agents negotiating goods and services, and a facilitator, are modeled and implemented.

In the experiments, agents are using different negotiation strategies, deadlines, and objects to buy or sell. A number of performance measures, such as utility, gain, number of successful negotiations, learning capabilities, are determined.

2.5.2. The Negotiation Protocol

There are different types of protocols, which are developed for different types of negotiation. The main protocols are described below:

a) **Contract Net Protocol** was introduced for distributed problem solving [Smith R.G., 1980]. It is used for task allocation problems. Using this protocol, agents negotiate about tasks. One agent, which is interested to perform a task, announces

other agents that a task is available. The agent might not be capable of performing the task on its own or it might try to find other agents that are able to perform the task more effectively. The agents that are interested in the task submit bids to the manager, which awards the task to the agent that sent the most satisfactory bid. Then, the agent starts the task. The bidding, bid processing and task processing phases are dependent on the problem.

b) **Rubinstein's Alternating Offers Protocol** [Osborne M.J. and Rubinstein A., 1994; Osborne M.J., 2004], in which two agents negotiate by taking actions at discrete time steps. In each time step, one agent makes a proposal to the other agent, which can accept or reject the proposal. If it accepts, negotiation ends, otherwise, the other agent makes a proposal at the next time step. This protocol is illustrated by the state transition diagram in Figure 2.7. [Wooldridge M., 2009].

c) **Monotonic Concession Protocol** [Rosenschein J.S. and Zlotkin G., 1994] is a particular case of Rubinstein's alternating offers model. This protocol forces an agent to make a concession to the other agent at each time step. It is guaranteed to stop, but it requires that agents know each other's utility functions, which is impossible in practice.

d) **Auctions** are particular negotiation protocols used for multilateral negotiations. Agents bid for items and special agents called auctioneers evaluate bids and allocate items. There are many types of auctions, the most important auction types being the English auction, the Dutch auction, first-price sealed-bid, second-price sealed-bid, and combinatorial auctions.

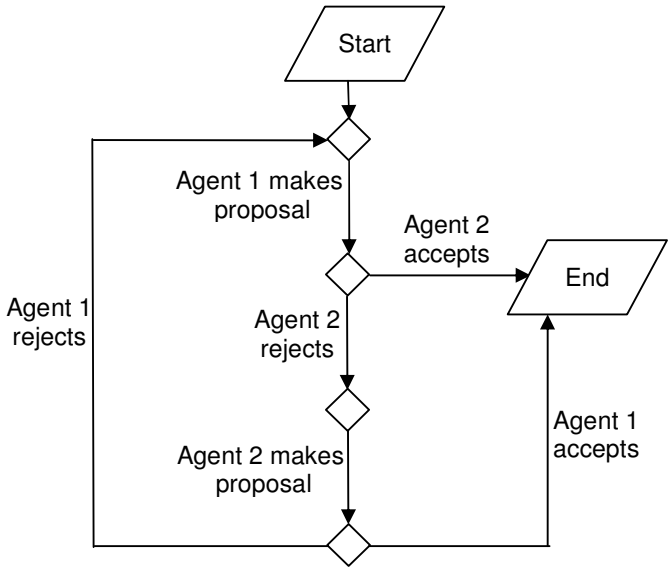


Figure 2.7. Alternating Offers Protocol [Wooldridge M., 2009]

In the evaluation of the results for negotiation protocols, there are **certain parameters** that can be used to measure different protocols [Kraus S., 2001]:

a) **Negotiation Time** - negotiations which end without delay are preferred over negotiations which are time-consuming. It is assumed that a delay in reaching an agreement causes an increase in the cost of communication and computation time spent on the negotiation. It is necessary to prevent the agents from spending too much time on negotiations resulting in deviation from their schedules for satisfying their goals;

b) **Efficiency** - an efficient outcome of the negotiations is preferred. An outcome increases the number of satisfied agents from the negotiation results. So, it is preferred that the agents reach **Pareto optima**l agreements. A deal is Pareto optimal if there is no other agreement that dominates it, for instance, there is no other accord that is better for the same agents and not worse for the others. In addition, if there is an agreement that is better for all the agents than terminating the negotiation, then it is preferred that the negotiation will end with an accord;

c) **Simplicity** - negotiation processes that are simple and efficient are better than complex processes. A simple strategy means that it is feasible to be built into an automated agent. A simple strategy supposes also that an agent will be able to compute the strategy in a reasonable amount of time;

d) **Stability** - a set of negotiation strategies are stable if, given that all the other agents included in the set are following their strategies, it is useful for an agent to follow its strategy. Negotiation protocols which have stable strategies are more useful in multi-agent environments than protocols which are unstable. If there are stable strategies, it is recommended to all agent designers to build relevant strategies into their agents;

e) **Money transfer** - money transfer may be used to solve conflicts. For example, a server may “sell” a data item to another server when relocating this item. This can be done by providing the agents with a monetary system and with a mechanism for secure payments. Since maintaining such a monetary system requires resources and efforts, negotiation protocols that do not require money transfers are preferred.

The main characteristics of the negotiation protocol are [Zlatev Z. et al., 2004]:

a) Each agent is making proposals corresponding to its own goal. A proposal contains an offer corresponding to the negotiation object, for instance a specific price, together with supporting information representing conditions under which this offer is made. Different offers have different supporting information, for example a goal to buy at a low price can contain several prices as possible offers, each of them being supported by the appropriate information. So, the negotiation object may be extended and may cover several issues related to the initial issue, for example terms and conditions under which an agent could accept a specific price;

b) The first of the two negotiating agents, which is unable to produce a new offer with supporting information for its goal, cancels it and searches for supporting information, if any, under which it can accept the counterproposal of the other agent. For example, a seller agent unable to find another way to support offers with high

prices considers selling at a low price and looks for supporting information under which it may be able to do so;

c) In such a case, the negotiation enters a conciliation phase and if the receiver agent can sustain the proposed supporting information, the negotiation ends with agreement on this offer and the supporting information accumulated so far. Otherwise, the sender takes this into account and tries again to find another way to support the goal of the agent. If this is not possible, then the negotiation ends in failure.

In automated negotiation, the combination of the software agents, representing the parties in the negotiation, and the negotiation protocol, establishing the interactions between these software agents, forms the automated negotiation system [Tamma V. et al., 2005]. The preferences over the negotiation object and the negotiation object itself are only input to this system and are not object to choices of the system user. The configuration of the automated negotiation system, which is the choice of software agents and interaction protocol used to perform automated negotiation, can be determined by the system users. So, while the negotiation object and the preferences of the users over this negotiation object are taken as they are, users can decide about the configuration and parameterization of the system employed to handle this negotiation problem by means of automated negotiation.

The interaction protocol builds the basis for the communication between software agents in an automated negotiation system. In some cases, the negotiations are finished earlier, most often by the protocol, if a deadline, in terms of a maximal number of turns, is reached before agents come to an agreement. Only in some of these cases, the agents have the choice to quit negotiations, and this decision is often based on whether or not a specified deadline is reached. Continuous concession strategies are applicable for automated negotiation in an open environment, as they can easily adapt to new problems and are independent of the opponent agent. Therefore, the idea is to apply protocols that are suitable for a population of software agents that follow these strategies. Such protocols should exhibit features that enable the software agents to interrupt their continuous concession strategy, if they think it is in their interest, which is to avoid exploitation or unfavorable outcomes [Filzmoser M., 2010].

When the negotiation begins, the protocol calls the agents in their initiation mode. The protocol provides the software agents, registered with the system as representing their parties, with the negotiation object indicated by the parties as input, as well as utility values for possible solutions. The agents create private storage variables, where they keep this information, together with information on the negotiation process. After this initiation of the software agents, the negotiation protocol chooses one agent and sends to him a call for proposals.

The software agent that receives the call for proposals from the interaction protocol is the first to make an opening offer and the one sending messages at odd turns during the negotiation process. The other software agent sends his opening

offer in the second turn and thereafter sends messages in the even turns of the negotiation process. As the messages are not exclusively offers, this protocol is called an alternating turn protocol, in which the agents alternate in taking their turns. Software agents can send one out of a set of messages determined by the protocol: **offer**, **reject**, **agree**, or **quit**. While a message of the type offer proposes one of the possible settlement of the negotiation object as agreement, which the opponent can accept or not, the other three message types are necessary for controlling the negotiation process.

An offer message constitutes a proposal for settling the negotiation. In human negotiations, there are many different kinds of messages besides offers, like threats, provision of or request for information, all aiming to influence the final outcome of the negotiation. However, it is argued that, while other messages are important, offers are the main type of messages in negotiations, as they promote proposals for the settlement. In order to be a proposal for settling the negotiation, an offer has to provide options for all issues of the negotiation object. The interaction protocol not only demands full package offers due to this, there are other protocols that work on an issue-by-issue basis or allow partial package offers, but the emphasis is on the opportunities to reach mutually beneficial agreements through package offers, which allow to tradeoff issues of lower importance against issues of higher importance.

If a software agent sends a reject message, it means that in this turn it does not propose a new offer, or make any other changes to the current state of the negotiation, but insist on its last offer. This message represents a strategy to avoid unfairly small concessions and exploitation by the opponent. Sending a reject message enables the software agent to discontinue the offer generation strategy it normally follows, according to which it would make some predetermined offer, for some reasons, without necessarily terminating the negotiation [Lin R. et al., 2009 a].

Several bargaining games in non-cooperative game theory do not require the explicit acceptance of an offer, but to terminate the negotiation, if the offers of the parties are compatible, that is if for both agents the demanded utility of their offer is lower than or equal to their utility of the offer proposed by the opponent. If the offers do not coincide, division rules can be used to split the surplus or choose one of the compatible offers as agreement. For this purpose, equal division of the surplus or some kind of arbitration, where either offer is chosen with equal probability, is usually applied. The problem of taking one out of two compatible offers by some arbitration rule mainly arises from the fact that in the above mentioned bargaining games agents have to make offers simultaneously. A software agent in the situation to send a message will first compare the opponent's last offer with his own offer to be sent next. If the opponent's offer affords higher or equal utility, the software agent accepts it, rather than proposing its next offer. So, the software agents in the sequential interaction protocol detect and use compatible offers themselves, rather than relying on the interaction protocol to do so. On the other hand, splitting procedures are only applicable if the zone of possible agreements is an area, rather than a set of points,

furthermore splitting does not definitely specify the actual outcome of the negotiation, as more than one of these possible settlements may afford the same utility to the parties. Therefore, splitting procedures, without additional refinements, are actually only applicable in a single-issue negotiation with continuous options for this single issue. Sending an agree message means that the agent accepts the last offer of the opponent as agreement. This clearly determines what the negotiation was settled for, as the last offer of the opponent is required to be a full package offer with options for all issues of the negotiation object.

Like in negotiations between humans, which do not need to end with an agreement [Beam C. and Segev A., 1997], an interaction protocol can allow the software agents to send a quit message. Sending this message fulfills one of the termination criteria of the interaction protocol and negotiations will end without agreement. Therefore, quit messages can be used by the software agents to break off negotiations, if they decide this is in their interest in the given context. For software agents following continuous concession strategies, the quit message is a mean to permanently interrupt the offer generation strategy the agent normally would follow, in not only denying to propose the next offer, as in case of the reject message, but in aborting to negotiate at all.

The interaction protocol terminates the negotiation either if:

a) a software agent sends an agree message, accepting the last offer of the opponent;

b) a software agent sends a quit message, breaking off the negotiation;

c) two subsequent messages of the two software agents were reject messages.

This last termination criterion is applied to avoid an infinite negotiation without progress towards an outcome, if this is an agreement or a break off for negotiation. When a software agent sends a reject message, this does not change its internal state, and means that the same message will also be sent in its next turn. If a message of the opponent causes state changes and, due to a different situation, the message of the software agent could be different from that of the previous round. If both agents send reject messages subsequently, there will be no state changes that lead to either agreement or break off of the negotiation anymore, but only an infinite number of alternating reject messages and the protocol terminates negotiations if this would occur. Repeating the same offer would have the identical effect as sending a reject message, representing no changes in the state of the negotiation. The offer generation strategies are designed in a way that they do not repeat offers, but send reject messages to interrupt concession making. However, this repetition of offers could easily be detected by the interaction protocol and treated the same as if a reject message was sent, or software agents could be allowed to repeat the previous offer, rather than to propose a new offer and the interaction protocol terminates negotiations if the two software agents subsequently repeated their last offers.

2.5.3. Automated Negotiation Agent Strategies

The strategies of most software agents, used in automated negotiation, are based on evolutionary computing, learning mechanisms, or time-dependent concession functions. These software agents are not readily applicable to actual automated negotiation, for an implementation in operative systems, for various reasons. First, time-based concession functions are not suitable due to the fast proceeding and therefore time insensitivity of automated negotiation, and the aim to improve outcomes over those reached by the human negotiation behavior these strategies imitate. The impatience of the negotiator or the costs associated with the mechanism of negotiation are good arguments for the decrease of the utility level demanded over time, which is the way time-based strategies are actually modeled.

The possible variety and complexity of negotiation problems and opponent strategies in automated negotiation causes problems for software agent strategies based on evolutionary computing or learning mechanisms. Software agents for automated negotiation can easily be programmed by or for human users. The media openness for automated negotiation, for new software agents acting on behalf of users with various preferences and for many different negotiation problems, has to be considered, when designing decision making algorithms. Software agent strategies based on evolutionary computing or learning algorithms are not flexible and generic enough to cope with this variety of possible new opponents, negotiation objects, and therefore transaction problems, determined by the various preferences over various objects. Evolutionary computing-based strategies, especially those implementing sequential threshold rules, might not have the chance to have the large number of interactions with the same opponent and for the same negotiation problem. These software agents need to reach good agreements by means of co-evolution [Beam C. and Segev A., 1997; Tu M.T. et al., 2000], and models of the opponent held by learning strategies might be inadequate for the variety of new opponent strategies.

Due to these concerns about existing approaches to determine the software agents' decision making algorithms, the focus is on the class of continuous concession strategies. These rule-based and rather deterministic algorithms, neither model their opponent, nor try to learn something about the opponent's preferences or strategy, but are reusable for various negotiation problems with different opponents.

2.6. Syntactic Interoperability

XML (Extensible Markup Language) standards are used as a way of providing a common syntax for exchanging heterogeneous information. So, in order to interchange information between both trading partners, it is necessary to define documents based on XML.

XML is the basis for integrating data within an enterprise and across supply chains, substantially reducing the cost of information interchange. In B2B area, there are different standards for describing interchanged business documents based on XML standard.

XML is a self-describing, text-based structured data format. The power of XML lies not so much with XML itself, but with the set of available tools for working with it. Web browsers can visualize it and commercial XML editors are available. The **XSLT** (Extensible Stylesheet Language Transformations) pattern language can be used to write simple scripts that transform an XML document into a new XML document in a different format, or into a non-XML document. Many high quality parsers and APIs are available for working with XML from Java programs [Friedman-Hill E., 2003].

XML is not a replacement for **HTML** (HyperText Markup Language). XML and HTML were designed with different goals. XML was developed to transport and store data, with focus on what data is, while HTML was designed to display data, with focus on how data looks. HTML refers at displaying information, while XML focus on carrying information.

The tags in XML are created by the author of the document. XML language has no predefined tags and allows the author to define the tags and the document structure. The tags used in HTML are predefined. HTML documents can only use tags defined in the HTML standard.

XML documents must contain a root element. This element represents the parent of all other elements. The elements in an XML document form a document tree. The tree starts at the root and branches to the lowest level of the tree. All elements can have sub-elements. Parent elements have children. Children on the same level are called siblings. All elements can have text content and attributes, as in HTML.

XML is extensible because, unlike HTML, the markup symbols are unlimited and self-defining. XML is a simpler subset of **SGML** (Standard Generalized Markup Language), the standard describing how to create a document structure. HTML and XML can be used together in many Web applications. XML markup may appear in a HTML page.

While many syntactic problems associated to the information interchanged in B2B have been solved using the XML standard, the notion of semantic interoperability still has problems. The difficulties associated to semantic interoperability occur due to the context dependency, therefore, it can only be understood in the context of its original source and purpose. A way to overcome this problem is to employ an explicit context model that can be used to re-interpret information in the context of a new information source and a new application. The use of ontologies is a promising approach, in order to show contextual information and to make a semantic preserving translation possible.

An ontology is a way of categorizing objects, such that they are semantically meaningful to a software agent. Ontology defines the relationship between similar products [Kholief M. et al., 2012].

To face the new global market, enterprises need to carry out collaborative relationships with their suppliers and customers on the Internet. However, the main obstacles are the syntactic and semantic interoperability at information level. XML technologies allow the syntactic interoperability.

An effective collaborative relationship between trading partners will not be successfully, until an environment to support the semantic integration will be developed. Some ontology based tools for solving the semantic interoperability between heterogeneous information systems could help. However, these tools do not support the entire ontology lifecycle or do not provide a collaborative environment. A support for collaborative engineering to maintain the shared ontology is the basic requirement. Furthermore, the ontology environment has to support the language interoperability.

Chapter 3. Knowledge Representation and Learning

3.1. Knowledge Representation for Negotiation Strategies

3.1.1. Knowledge Representation Using Rules

A framework for agent negotiations should contain the negotiation infrastructure, the generic negotiation protocol, and the negotiation rules and strategy.

The negotiation infrastructure defines roles of negotiation participants and of a facilitator. Participants negotiate by exchanging proposals. Depending on the negotiations type, the facilitator can also play the participant role.

The negotiation protocol defines the three phases of a negotiation: admission, exchange of proposals, and agreement formation, in terms of how and when messages should be exchanged between the facilitator and the participants.

Negotiation rules are used for enforcing the negotiation mechanism. Rules are organized into a taxonomy [Badica C. et al., 2005]:

- rules for participants' admission to negotiations;
- rules for checking the validity of negotiation proposals;
- rules for protocol enforcement;
- rules for updating the negotiation status and informing participants;
- rules for agreement formation;
- rules for controlling the negotiation termination.

In a rule-based system, the inference engine controls the process of applying the rules to the working memory, in order to get the results of the system. An inference engine works in cycles, as described below [Friedman-Hill E., 2003]:

a) All the rules are compared to the working memory to decide which ones should be activated during this cycle. This unordered list of activated rules, together with any other rules activated in previous cycles, forms the conflict set.

b) The conflict set is ordered to form the agenda, which is the list of rules whose right-hand sides are executed. The process of ordering the agenda is called conflict resolution. This strategy for a given rule engine depends on many factors, only some of which are under the programmer's control.

c) In order to complete the cycle, the first rule on the agenda is fired, possibly changing the working memory, and the entire process is repeated. This repetition implies a large amount of redundant work, but many rule engines use different techniques to avoid most or all of the redundancy. In particular, results from the pattern matcher and from the agenda's conflict solver can be preserved across cycles, so that only the essential, new work needs to be done.

3.1.2. JESS and RuleML

JESS (Java Expert System Shell) is a rule-based system, implemented in Java language. It was developed starting from the expert system **CLIPS** (C Language Integrated Production System), but it evolved into a complete and distinct rule-based system [Florea A.M. et al., 2008].

Using Jess, it is possible to write applets and Java applications, which have the capacity to reason, using the knowledge from the declarative rules. Jess offers easy integration with other Java based software.

An expert system has a set of rules, which could be applied many times on a set of facts. Jess uses a very efficient algorithm, called Rete, in order to unify rules and facts.

An expert system based on Rete algorithm constructs a set of nodes, where each node, with the exception of the root, refers to a pattern, which is present in the left hand side of a rule. The path from the root node to a leaf node defines completely the left hand side of a rule. Each node memorizes the facts which satisfy that pattern. When new facts are added or modified, these are propagated along the network, labeling the nodes when the fact matches with that pattern. When a fact or a combination of facts does all the patterns for a given rule to be fulfilled, a leaf node is reached and the respective rule is executed.

So, Jess uses the fast and efficient Rete algorithm for pattern matching. The strength of Rete is that it uses a set of memories to keep information about the success or failure of pattern matches during previous cycles. The Rete algorithm involves building a network of pattern matching nodes. Jess uses many different kinds of nodes to represent the different kinds of pattern matching activities [Friedman-Hill E., 2003].

Jess offers the basic elements of an expert system [Florea A.M. et al., 2008]:

a) the list of facts and the list of instances – the global memory for data;

b) the knowledge base – contains all the rules, that is the rule base;

c) the inference engine – controls the rules' execution.

A program written in Jess can contain rules, facts and objects. The inference engine controls which rules should be executed and when they should be performed. A rule-based expert system, written in Jess, is a data driven program, in which facts and objects represent data which generated the execution, using the inference engine.

Jess executes always the actions from the right hand side of a rule with the highest priority. After that, the rule is removed and the next rule is executed, in decreasing order of priorities.

Jess offers two different ways to solve the conflicts: depth-first (LIFO – Last In First Out) and breath-first (FIFO – First In First Out). In the case of depth-first strategy, the most recently activated rules are used, before the rules activated not so recently and which have the same priority. In the case of breath-first strategy, the

rules with the same priority are applied, in the order in which they have been activated. It is difficult to decide which strategy is better, without considering the specific application.

In Jess, modules allow a knowledge base to be partitioned. Every construct defined must be placed in a module. The programmer can explicitly control which constructs in a module are visible to other modules and which constructs from other modules are visible to a module. The visibility of facts between modules can be controlled in a similar manner. Modules can also be used to control the flow of execution of rules.

Jess stores the contents of the working memory using a set of customized indices that make looking up a particular piece of information very fast. Even though Jess uses a data-centric index internally, the view of the working memory looks like a simple list. Each item in the working memory appears on this list in the order in which it was added [Friedman-Hill E., 2003].

The combination of forward-chaining inference rules and backward-chaining infrastructure rules is a powerful and common pattern in Jess systems.

RuleML (Rule Markup Language) is an open language, based on rules **XML**/**RDF** (Extensible Markup Language / Resource Description Framework). This allows the exchange of rules between different systems, including distributed software components on the Web and heterogeneous client-server systems. RuleML offers the XML syntax for interoperable knowledge rules representation between the main rules systems.

RuleML contains a hierarchy of rules, starting from reaction rules (event-condition-action rules), integrity constraint rules, derivation rules, up to facts (derivation rules without premises).

The rule hierarchy in RuleML constitutes a partial ordering, on the first level being the reaction rules. The second level contains the integrity constraint rules and the derivation rules. The third level specialises the derivation rules into facts, as shown in Figure 3.1.

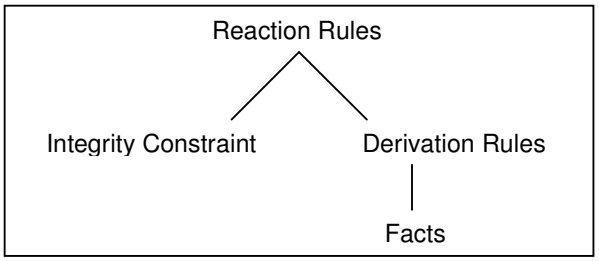


Figure 3.1. RuleML Rules Hierarchy

3.1.3. Knowledge Representation using Ontologies

An ontology can be viewed as a formal representation of a set of concepts from a certain domain, and also of the relations between these concepts.

The main reasons for using ontologies are [Noy N.F. and McGuinness D.L., 2001]:

a) to have a common understanding about the information structure – suppose that a set of Web sites contains medical information and delivers medical services of e-commerce type [Serbanati L.D. and Radu S., 2013]. If these sites have as basis the same ontology for their terms, then the Web agents can extract and compose the information from these different sites. The agents can use this compound information to answer the user queries or as input data to other applications;

b) to allow the knowledge reuse from the domain – the models from different domains must have a representation for the notion of time, which includes the notions of time interval, moments in time, time measurement units. If such an ontology is already developed, it can be reused in different domains. If it is necessary to develop a complex ontology, then it is possible to integrate the existing ontologies, which describe parts of the domain;

c) to express the hypotheses used in that domain – it is possible to easily change these hypotheses, when the knowledge of the domain is changing;

d) to separate the knowledge of the domain from its implementation;

e) to analyze the domain knowledge – the formal analysis of terms is very useful when reusing or extending ontologies.

Developing an ontology is like defining a data set and its structure, in order to be used by a program. An ontology in an explicit formal description of the concepts from a domain: classes of the properties of each concept, which describe different features and attributes of the concepts (also called roles), and attribute characteristics or restrictions (implications, also called role restrictions).

3.1.4. OWL Language

OWL (Web Ontology Language) is a language used to represent ontologies. OWL is an extension of RDF Schema. Data described by an OWL ontology are interpreted as a set of figures and a set of properties, based on which the figures are related.

An OWL ontology consists of a set of axioms, based on which constraints are associated to the group of figures and types of relations allowed between them. These axioms give the semantics which permits the system to deduce the additional information, based on data given explicitly.

OWL specification includes the definition of three OWL components [Horrocks I. et al., 2003]:

a) **OWL Lite** was developed to support hierarchical classifications and simple constraints. The restrictions cardinality could be 0 or 1, and it is possible to specify a part or all the values of the property upon which the restriction is applied. Also, it is possible to specify classes or properties which are equivalent (it is very useful to specify the fact that two concepts in different ontologies represent the same thing), and the fact that the properties could be functional, symmetric, transitive or inverse;

b) **OWL DL** is a more advanced language, based on a decidable subset of description logic. There are possible relations on disjunctions of classes or union, intersection or complement between classes. Also, it is possible to specify cardinalities for restrictions in the form of any natural number;

c) **OWL Full** is based on the semantics of OWL Lite and OWL DL and was built to keep some compatibilities with RDF Schema. OWL Full allows an ontology to develop the meaning of the predefined vocabulary (RDF or OWL).

Each of these sub-languages is a syntactic extension of its predecessor, that is every OWL Lite ontology is also an OWL DL valid ontology, and every OWL DL ontology is also an OWL Full valid ontology.

Domain ontologies describe the vocabulary related to a generic area. These model a certain domain or a part of the world. They represent the particular sense for the terms which are applied to the respective domain.

Level ontologies describe very general concepts, such as: space, time, matter, object, event, action, which are independent from a particular problem or from a domain. Such an ontology is a model regarding the common objects which can be applied in different domain ontologies.

3.2. Learning Algorithms

3.2.1. ID3 Algorithm

In decision tree learning, **ID3** (Iterative Dichotomiser 3) is an algorithm used to generate a decision tree [Murthy S.K., 1998]. It prefers smaller decision trees (simpler theories) over larger ones. However, it does not always produce the smallest tree, and is therefore a heuristic.

Each level of the tree is associated to one attribute. Alternate branches starting from one node are labeled with the values of the attribute associated to that level. The leaves of the tree are tagged with the classes associated to the objects to be classified.

First, the decision tree is built and then it is used on unknown instances, in order to classify them.

Each branch associated to a value of the attribute is labeled with all the examples that have the same value for the attribute. This process continues on each level associated to an attribute, until a node is obtained, for which the associated

examples are all in the same class. This node becomes a leaf and is labeled with that class.

Suppose there is a set of training instances **C**. The ID3 algorithm performs the following steps:

Step 1: **If** all the instances in **C** are positive **then** create **Yes** node and stop;
If all the instances in **C** are negative **then** create a **No** node and stop;
Else, select an attribute **A**, with values $\mathbf{v_1, \ldots, v_n}$ and create a decision node.

Step 2: Partition the training instances in **C** into subsets $\mathbf{C_1, C_2, \ldots, C_n}$, according to the values of **A**.

Step 3: Apply the algorithm recursively to each of the sets $\mathbf{C_i}$.

Each attribute of an instance is considered to have a certain informational contribution for classifying that instance. The heuristic of the ID3 algorithm measures the informational gain brought by each attribute and chooses to test in the root of the decision tree the attribute **A**, which maximizes the informational gain.

In order to measure the informational content from a message, the information theory is used. A message is regarded as an instance from a set of all possible messages. Sending a message is equivalent with selecting a certain message from this set. The informational content of a message depends on the set size and the frequency of each message. The informational content of a message is defined as the probability of occurrence for any possible message. Having a set of messages: $\mathbf{M = \{m_1, m_2, \ldots, m_n\}}$ and a probability $\mathbf{p(m_i)}$ of occurrence for each message, the informational content of a message from **M** is defined by:

$$I(M) = \sum_{i=1}^{n} - p(m_i) * \log_2(p(m_i)) \tag{3.1}$$

The ID3 algorithm uses the information theory in order to select the attribute, which gives the highest informational gain, when classifying the instances from the training set. A decision tree is considered as having information about the classification of the examples from the training set. The informational content of the tree is computed using the probabilities for different classifications.

For a certain attribute **A**, the informational gain obtained by selecting this attribute as the root of the decision tree is equal to the total informational content of the tree minus the informational content necessary to finish the classification (building the tree), after selecting the attribute A as the root of the tree:

$$G(A) = I(T) - E(A) \tag{3.2}$$

The informational quantity needed to finish constructing the tree is the weighted average of the informational content from all the subtrees. The weighted average is computed by multiplying the informational content for each subtree with the instances percent from that subtree, and then summing up these products.

Suppose that there is a set of training instances **C**. If the attribute **A** with **n** values is put in the root of the tree, this determines the partition of the set **C** into the

subsets **{C_1, C_2, …, C_n}**. The estimation of the informational quantity needed to construct the decision tree, after the attribute **A** was chosen as the root, is given by:

$$E(A) = \sum_{i=1}^{n} \frac{|C_i|}{|C|} * I(C_i) \tag{3.3}$$

3.2.2. C4.5 Algorithm

C4.5 algorithm represents an extension of the ID3 algorithm, which takes into account the possible input and output noise, works with inadequate attributes, and could have unknown values for an attribute [Mazid M.M. et al., 2010].

C4.5 algorithm constructs decision trees from a set of training data in the same manner as ID3. Any correct decision tree will classify the objects proportionally with their representation in the set of training instances. When a decision tree is used to classify unknown objects, it will return a certain class.

In the case when there are no objects which has a certain value for an attribute, the ID3 algorithm labels the leaves with **null** or **failure**, so it is not possible to make the classification. As an improvement, the C4.5 algorithm generalizes and assigns to the leaf the class which appears most frequently in the set of training instances.

The errors which could appear in the training set give raise to inadequate attributes and to decision trees with higher complexity than necessary. If there are missing attributes in the training set, it is assigned the value, which appears with the highest frequency, or there are used probabilities to determine the distribution probability for the values of the attribute **A** in the training set **C**, depending on the membership to a class:

$$prob(A = A_i \mid class = P) = \frac{prob(A = A_i \wedge class = P)}{prob(class = P)} \tag{3.4}$$

and it is chosen the value with the highest probability.

Another solution used by the C4.5 algorithm, when evaluating the informational gain, is to assign to unknown value objects, values distributed over the values of the attribute, proportional with the relative frequency of these in the training set. The unknown values will decrease the informational gain.

When the attribute **A** has many symbolic values, the separation information is used. This is the information quantity necessary to determine the value of an attribute **A** in a learning set **C**. Let be $\mathbf{P_{AC}}$ the distribution probability for the values of the attribute **A**:

$$P_{AC} = (\frac{|A_1|}{|C|},...,\frac{|A_v|}{|C|}) \tag{3.5}$$

The separation information (**SI**) measures the informational content of the answer of the value for the attribute **A**:

$$SI(A) = -\sum_{i=1}^{v} \frac{p_i + n_i}{p + n} * \log_2 \frac{p_i + n_i}{p + n} \tag{3.6}$$

The improvements of the C4.5 algorithm with respect to the ID3 algorithm refers at dealing with continuous and discrete attributes, processing training data which have missing attribute values, taking into account attributes with different costs, and pruning trees after creation.

3.2.3. Reinforcement Learning

Reinforcement learning (**RL**) is the problem faced by an agent that learns behavior through trial-and-error interactions with a dynamic environment [Kaelbling L.P. et al., 1996].

There are two main strategies for solving reinforcement learning problems. The first is to search in the space of behaviors in order to find one that performs well in the environment. This approach has been taken in genetic algorithms and genetic programming. The second strategy is to use statistical techniques and dynamic programming methods to estimate the utility of taking actions in states of the world. In the standard reinforcement learning model, an agent is connected to its environment via perception and action, as presented in Figure 3.2 [Kaelbling L.P. et al., 1996].

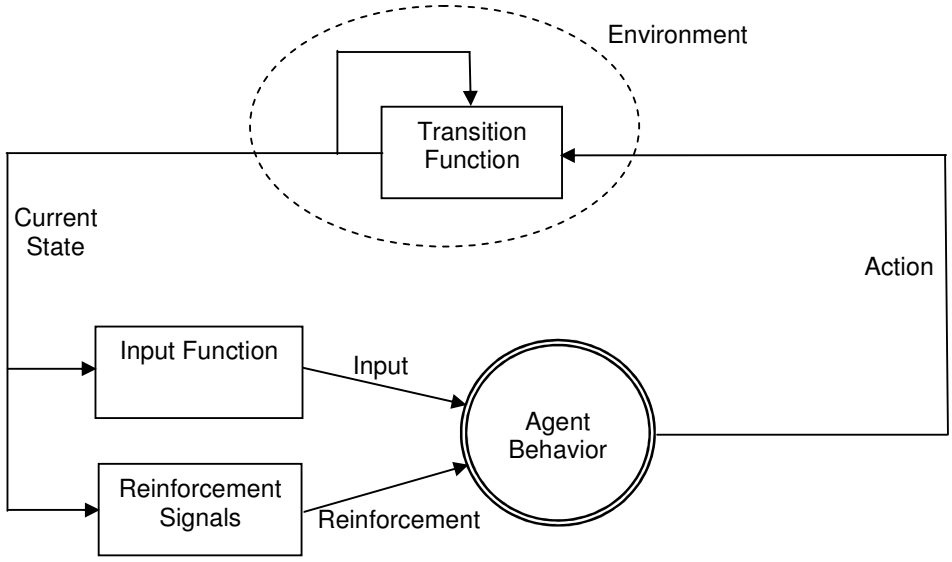


Figure 3.2. Standard Reinforcement Learning Model [Kaelbling L.P. et al., 1996]

On each step of interaction, the agent receives as input, **i**, some indication of the current state, **s**, of the environment. The agent then chooses an action, **a**, to generate as output. The action changes the state of the environment, and the value of this state transition is communicated to the agent through a scalar reinforcement signal, **r**. The agent's behavior, **B**, should choose actions that tend to increase the long-run sum of values of the reinforcement signal.

The RL model consists of a discrete set of environment states, **S**, a discrete set of agent actions, **A**, a set of scalar reinforcement signals, **R**, an input function **I**, which determines how the agent views the environment state; it is assumed that it is the identity function (that is, the agent perceives the exact state of the environment), and an environment transition function, **T**.

The value of a state is the total sum of a reward, which an agent hopes to gain in the future, starting from a certain situation. The value of a state can be seen as the long run reward prediction. States with high values are searched, not with high rewards. The reward is given directly from the environment, but the value should be estimated by the agent, through experience. The agent should find a policy $\pi: S \rightarrow A$, which is a function connecting states with actions, maximizing a measure of the long run reward. The agent should learn an optimal behavior, which is a policy that produces the maximum estimated value. The optimal policy is denoted π^*.

There are different behavior models for the agent. In the finite **horizon model**, the agent go through a number of **n** states and reaches the final state, this representing an episode, then the process is repeated. The estimated reward **U** is computed with respect to the reward received in the state S_t in the future:

$$U(S_t) = \sum_{t=0}^{n} R(S_t) \tag{3.7}$$

In the **infinite horizon model**, the agent lives forever and optimizes the long run reward:

$$U(S_t) = \sum_{t=0}^{\infty} R(S_t) \tag{3.8}$$

The **discounted infinite horizon model** optimizes the long run reward, but future rewards are decreased with a discount factor δ, where $0 \leq \delta < 1$:

$$U(S_t) = \sum_{t=0}^{\infty} \delta^t * R(S_t) \tag{3.9}$$

3.2.4. Q-Learning

Q-learning is a reinforcement learning technique that works by learning an action-value function that gives the expected utility of taking an action in a certain state and then following a fixed policy [Manju S., Punithavalli M., 2011]. One of the strengths of Q-learning is that it is able to compare the expected utility of the available actions, without requiring a model of the environment.

The model consists of an agent, states **S** and a set of actions per state **A**. By performing an action $a \in A$, the agent can move from a state into a next state. Each state provides the agent a reward. The goal of the agent is to maximize its total reward. It does this by learning which action is optimal for each state.

In Q-learning, instead of utilities, action-value pairs are learned. The function **Q** assigns an estimated utility, when executing an action in a certain state: $Q: A*S \rightarrow U$.

Q(a,s) represents the discounted estimated sum of the future rewards, obtained starting from state **s**, choosing action **a** and following an optimal policy:

$$Q(a,s) \leftarrow Q(a,s) + \alpha * (R(s) + \delta * \max_{a'} Q(a',s') - Q(a,s)) \quad (3.10)$$

This is computed after each transition from state **s** to state **s'**, where **δ** is the discount factor and **α** is the learning rate. The formula can be rewritten as:

$$Q(a,s) \leftarrow Q(a,s) * (1-\alpha) + \alpha * (R(s) + \delta * \max_{a'} Q(a',s')) \quad (3.11)$$

3.2.5. Learning Classifier Systems

Learning classifier systems are a kind of rule based systems, which have general mechanisms for parallel processing rules, for adaptive generation of new rules and for effectively testing the existing rules [Bull L., 2004]. Learning classifier systems are machine learning systems, which are based on reinforcement learning and genetic algorithms.

Learning classifier systems can be divided into two types, with respect to the way in which the genetic algorithms are applied. The first type is represented by **Pittsburgh learning classifier system**, which has a population of separate set of rules, in which the genetic algorithms recombine and give the best from the sets of rules. The second type is described by the **Michigan learning classifier system**, in which there is only one population and the algorithm action is concentrated on selecting the best classifiers from the rules set. The Michigan type learning classifier systems has two possibilities to define the fitness: strength-based (ZCS) and accuracy-based (XCS).

A **strength-based classifier** (**ZCS**) could be represented as a triple **<c,a,p>**, where **c** is the condition, **a** is the action, and **p** is an estimation of the expected reward, which an agent could receive if it uses this classifier. In this classifier type there is no message list and the Q-learning algorithm is used in the learning process.

An **accuracy-based classifier** (**XCS**) has a different form of rules, instead of condition → action, the rules has the form condition, action → effect. By effect it is understood the expected effect (the next state) corresponding to the action.

A learning classifier system is an adaptive system, which learns to perform the optimal action, which will receive the best reward from the environment. The rules evolve based on a match set, which is the set of classifiers satisfying the current state. Using a mechanism based on fitness, the action to be executed is selected.

Chapter 4. Adaptive Negotiation Strategies

4.1. The Negotiation Mechanism

Designing a negotiation mechanism means defining a negotiation protocol and a negotiation strategy for the agents in the system. The choice of the negotiation protocol in conjunction with the negotiation strategies adopted by the agents determines the type of outcome that is produced by the mechanism. A protocol used for bilateral negotiations is the monotonic concession protocol [Rosenschein J.S. and Zlotkin G., 1994].

A buyer agent can negotiate over multiple issues in parallel and, for each issue, the agent concurrently negotiates with its trading partners. If the buyer doesn't know how to set the price of each of the issues, one approach is to negotiate over all the issues in parallel [An B. et al., 2010]. For each issue, there are multiple trading partners and the agent concurrently negotiates with all of them. Generally, a buyer obtains more desirable negotiation outcomes, when it negotiates concurrently with all the sellers in competitive situations, in which there is information uncertainty and there is a deadline for the negotiation to complete. Inefficiency may arise in sequential negotiation, when considering the overall time cost to complete all the necessary negotiations [Fatima S.S. et al., 2006].

As agents can choose to decommit from agreements, negotiation consists of a bargaining stage and a decommitment step for each negotiation thread. A pair of buyer and seller agents negotiates by making proposals to each other. At each round, one agent makes a proposal. Many buyer-seller pairs can bargain simultaneously, since each pair is in a negotiation thread. If the seller accepts the proposal of the buyer, negotiation terminates with a tentative agreement. If the seller rejects the proposal of the buyer, negotiation terminates with no agreement. If the seller makes a counterproposal, bargaining proceeds to another round and the buyer can accept the proposal, reject the proposal, or make a counterproposal.

Bargaining between two agents terminates: when an agreement is reached or with a conflict (no agreement is made), when one of the two agents' deadline is reached or one agent quits the negotiation. After a tentative agreement is made, an agent has the opportunity to decommit from the agreement and it pays the penalty to the other party involved in the decommited agreement [Sandholm T. and Lesser V., 2001].

The multi-agent system involved in negotiation should have the following properties [Kumar S., 2012]:

a) The individual agents in the system are autonomous, rational, and self-interested. These agents might be owned by different organizations, and therefore, they make independent decisions and attempt to maximize the expected utilities for their owners;

b) The agents encounter situations where they cannot satisfy their goals by themselves. They have to cooperate with other agents and arrive at mutually beneficial agreements in order to further their own selfish interests;

c) The agents have to cooperate in situations where there is a conflict of interest. Multi-agent interactions in the system can be modeled as constant-sum games, where an agent's gain is always at the expense of others in the system;

d) There can be an arbitrarily large number of agents in the system. However, each agent can interact with only a small subset of other agents in the system.

In a negotiation mechanism there are many desirable properties [Goradia H.J and Vidal J.M., 2007]. **Efficiency** is the most important property for a mechanism. By economic efficiency it is understood that the agreement mechanism yields should be close to optimal. **Optimality** can be measured in many ways, and is domain-specific. It is desirable a mechanism to be computationally efficient for an obvious reason: faster the agreement is reached, less time is consumed in negotiation. A mechanism that involves less communication between the agents during the negotiation process would be preferred in a real-world setting.

Another desirable property in a mechanism is **stability**. A mechanism is stable if it provides all agents with a desire to behave in a particular way. Even for many settings with cooperative agents, it is assumed that the agents work towards improving their individual utilities.

Distribution of command and decision-making are essential in certain settings, where the nature of the problem makes it infeasible to aggregate all the necessary data. Distribution is desirable also for systems that are not inherently decentralized, as it avoids a single point of failure and minimizes performance bottleneck.

Scalability is another important issue in settings where there are a lot of agents. The mechanism should not be affected by the increasing number of agents in the system.

Suppose there are many businesses in an electronic marketplace, each specializing in offering certain goods or services for a price [Wellman M.P. and Wurman P.R., 1998]. Their concern is maximizing their profits by asking the highest possible price for the goods or services they offer. These businesses may not be able to complete a task alone, because they do not possess all the capabilities needed to handle the task completely. In such cases, they have to collaborate with each other to fulfill customer requests in the marketplace.

4.2. Negotiation using Decision and Game Theory

A framework for one-to-many negotiation by means of conducting a number of concurrent coordinated one-to-one negotiations is presented in [Rahwan I. et al., 2002 a]. In this framework, a number of agents, all working on behalf of one party, negotiate individually with other parties. After each negotiation cycle, these agents report back to the facilitator, which evaluates how well each agent has done, and

issues new instructions accordingly. Each individual agent conducts reasoning by using constraint-based techniques. It is outlined that two levels of strategies can be used, the individual negotiation type, and the coordination level. It is also showed that the one-to-many negotiation architecture can be directly used to support many-to-many negotiations. In the prototype Intelligent Trading Agency (**ITA**), agents autonomously negotiate multi-attribute terms of transactions in an e-commerce environment tested with a trading scenario.

A multi-agent framework for a negotiation system that supports the evaluation of messages, the management of the negotiation messages, and the messages exchange among the negotiation agents is designed in [Choi H.R. et al., 2005]. It is dealing with a simple problem, where the agents exhibit common attributes that do not need a complex classification scheme. It means that their study focuses on strategy, rather than ontology, in the development of an automated negotiation system.

An adaptive bilateral negotiation between software agents in e-commerce environments is studied in [Narayanan V. and Jennings N. R., 2005]. The research supposes that the agents are self-interested, the environment is dynamic, and both agents have deadlines. Such dynamism assures that the agents' negotiation parameters depend on the state of the encounter and the environment. The study presents an algorithm, which the agents can use to adapt their strategies to modifications in the environment, to get a deal with respect to their deadlines and before the available resources for negotiation are exhausted. An adaptive negotiation model is defined as a Markov Decision Process. Using a value iteration algorithm, a possibility to find optimal policies for the negotiation problem is described, without explicit knowledge of the dynamics of the system. A representative negotiation decision problem using this technique is described and it is showed that it is a promising approach for analyzing negotiations in dynamic settings. With empirical evaluation, it is concluded that the agents using this algorithm learn a negotiation strategy that adapts to the environment and allows reaching agreements.

An adaptive approach in agent-based negotiation involving on-line prediction of the opponent behavior based on the parametric non-linear regression analysis is proposed in [Brzostowski J. and Kowalczyk R., 2006]. A decision-making mechanism, using the information obtained by the regression mechanism, is also proposed. The predictive decision-making mechanism for the negotiation agent is based on the history of offers in the current negotiation encounter. The approach proposed allows the negotiation agents to predict more complex behavior of the negotiation opponent, in terms of the mixture of its time-dependent and behavior-dependent tactics. They perform experiments in order to validate the proposed approach. The results show that the predictive decision-making gives better results in terms of the utility gains for the adaptive negotiation agent, as compared with a range of non-predictive negotiation strategies.

A multi-issue negotiation between self-interested autonomous agents is analyzed in [Fatima S.S. et al., 2007]. The agents have time constraints in the form of both deadlines and discount factors. There are **n** issues for negotiation, where each issue is viewed as a pie of size one. The issues are indivisible, that is the individual things cannot be split between the parties and each issue must be allocated entirely to an agent. The problem is for the agents to decide how to allocate the issues between themselves, such that to maximize their individual utilities. For such negotiations, the equilibrium strategies are obtained, for the case where the issues for negotiation are known a priori to the parties. Then, it is analyzed their time complexity and showed that finding the equilibrium offers in an **NP**-hard problem, even in a complete information setting. In order to overcome this computational complexity, approximately optimal negotiation strategies are presented, which are computationally efficient, and it is shown that they form an equilibrium.

Bilateral multi-issue negotiation is analyzed in [Fatima S.S. et al., 2010]. In this framework, the issues are divisible, there are time constraints in the form of deadlines and discount factors, and the agents have different preferences over the issues. The objects are negotiated using the package deal procedure. The set of issues to be negotiated represent a choice variable, that is the agents can decide what issues to bargain. This set is called the negotiation agenda. Since the negotiation outcome depends on the agenda, it is important to determine which agenda maximizes an agent's utility and is therefore the optimal one. In this approach, polynomial time methods are presented, for finding an agent's optimal agenda.

A comprehensive reasoning model for service-oriented negotiation is described in [Sierra C. et al., 1997]. This determines which potential servers are contacted, whether negotiation proceeds in parallel with all servers or whether it runs sequentially, what initial offers are sent out, what is the range of acceptable agreements, what counteroffers are generated, when negotiation is canceled, and when an agreement is reached.

A formal account of a negotiating agent's reasoning component is presented. The focus is on the processes of generating an initial offer, evaluating incoming proposals, and creating counterproposals. The model specifies the key structures and processes involved in this setting, and defines their inter-relationships. The model was shaped by practical considerations and insights, starting from the development of a system of negotiating agents for business process management. The main contributions of this work are: a) it allows flexible bargaining schemes to be defined; b) it is based on assumptions which are realistic for autonomous computational agents; and c) it presents some initial results on the convergence of negotiation.

4.3. Learning Algorithms used in Negotiation

A multi-agent system for supply chain management is described in [Chen Y. et al., 1999]. In this framework, functional agents can join in, stay, or leave the system. The Supply Chain Management System (**SCMS**) functionality is implemented through agent-based negotiation. When an order arrives, a virtual supply chain may emerge from the system through automated or semi-automated negotiation processes between functional agents. A framework is presented and a number of negotiation performatives are described, which can be used to construct pairwise and third party negotiation protocols for functional agent cooperation. It is also explained how to formally model the negotiation process by using Colored Petri Nets (**CPN**) and it is provided an example of establishing a virtual chain by solving a distributed constraint satisfaction problem.

The multi-agent meeting scheduling problem in defined in [Crawford E. and Veloso M., 2005], in which distributed agents negotiate meeting times on behalf of their users. While many bargaining approaches have been proposed for scheduling meetings, it is not well understood how agents can negotiate strategically in order to maximize their users' utility.

In order to negotiate strategically, agents need to learn choosing good strategies for bargaining with other agents. The playbook approach, introduced for team plan selection in small-size robot soccer, can be used to choose strategies. Selecting strategies in this manner offers some theoretical guarantees about regret. The experimental results prove the effectiveness of the approach. The space of negotiation strategies is huge, and thus it is not possible for an agent to learn how to negotiate in the complete space. The plays-based approach cuts the strategy space down to a set of procedures that are effective in different situations, allowing an agent to learn which of these strategies work best with different fixed-strategy agents.

An adaptive one-to-many negotiation strategy for multi-agent coalition formation in dynamic, uncertain, real-time, and noisy environments is proposed in [Soh L.K. and Li X., 2004]. The strategy focuses on multi-issue negotiations, where each issue is a request from the initiating agent to the responding agent. The initiating agent conducts concurrent negotiations with responding agents and in each negotiation it employs a pipelined one-at-a-time approach, or a confidence-based, packaged approach. In the first case, lacking knowledge on the responding agent, it negotiates one issue at a time. In the second case, with confident knowledge of the past behavior of the responding agent, it packages multiple issues into negotiation. This adaptive strategy is incorporated into a multi-phase coalition formation model (**MPCF**), in which agents learn to form coalitions and perform global tasks.

The creation of effective and efficient negotiation mechanisms for real-world applications is a challenging problem, because negotiations in such a context are characterized by combinatorial complex negotiation spaces, tough deadlines, very limited information about the opponents, and volatile negotiator preferences. So,

practical negotiation systems should have effective learning mechanisms to obtain dynamic domain knowledge from the possibly changing negotiation contexts. Adaptive negotiation agents are presented in [Lau R.Y.K. et al., 2006], which have robust evolutionary learning mechanisms to deal with complex and dynamic negotiation contexts. The experimental results show that genetic algorithms-based adaptive negotiation agents outperform a theoretically optimal negotiation mechanism that guarantees Pareto optimality. This allows the development of practical negotiation systems for real-world applications. Genetic algorithm-based adaptive negotiation agents are empowered by the effective evolutionary learning mechanisms, such that these agents can learn the opponents' preferences gradually and continuously adapt to the changing negotiation contexts. The design of genetic algorithm-based adaptive negotiation agents fulfills all the requirements of practical negotiation systems, because these agents can simulate a wide spectrum of negotiation attitudes, identifying near optimal solutions, based on limited information about the negotiation spaces, continuously learning the opponents' preferences, and adapting to the changing negotiation contexts.

A negotiation model, which contains a set of self-interested cognitive agents, capable to reason on different issues about the object to be negotiated is proposed in [Florea A.M. and Kalisz E., 2008]. A model of heuristic negotiation between self-interested agents is presented, which allows the use of arguments, negotiation over multiple issues of the negotiation object, single and multi-party negotiation, and learning of the agent's negotiation primitives. This model uses negotiation objects and negotiation frames to separate the object of negotiation from the negotiation process. In order to negotiate strategically, the agents use a reinforcement learning algorithm applied on a specific state space representation of the negotiation process.

A Belief-Desire-Intention model of agents, required in order to support the extended set of primitives, is presented in [Bratman M.E., 1999]. In a BDI model, the agents are endowed with beliefs about the environment and the other agents in the environment, with intentions to execute actions structured into plans and desires, which represent the outcomes the agents want to achieve. A consistent subset of desires forms the agent goals, towards which plans should be developed. It is proposed a reinforcement learning approach that may permit the negotiator to learn which negotiation primitive to use in a certain state of negotiation.

4.4. Negotiation Systems using Argumentation

A model of negotiation between self-interested agents is described in [Florea A.M., 2002], that captures different negotiation situations and objects, including arguments in favor of successful contracts. The negotiation protocol is specified using two alternate representations: negotiation trees and negotiation definite clause grammars. A reinforcement learning algorithm is proposed, that can be used by the negotiator to learn its negotiation strategy, when faced with multiple negotiation

primitives. The reinforcement learning approach is based on utilities of negotiation objects and negotiation states.

The negotiation technique used is a combination of a heuristic approach with an argumentation-based negotiation. The negotiator A issues a request for action, or service to be performed, or service to be offered, the request being directed to agent B. The agent B may accept the request, may reject it, may modify the request by changing the value of an attribute of the negotiation object or by adding a new attribute. Negotiation may continue by performing several consecutive steps, in which one or the other agent modifies the negotiation object, a successful contract has been concluded or the negotiation failed.

A multi-agent system, in which agents are able to negotiate in order to satisfy their goals and desires, is presented in [Carabelea C., 2002]. The system is open, the agents in the system are self-interested and are using argument-based negotiation to reach agreements regarding cooperation and goal satisfaction. Negotiation is performed using different types of arguments, varying from quantitative ones, such as money or trade objects, to qualitative arguments, such as promises, appeal to past promises and past examples.

The objects being negotiated are virtual objects, which may represent physical objects, actions performed on their behalf, desires of other agents, other agents' preferences, or money. The argument-based negotiations are covering both economic type negotiations and symbolic daily life ones. The agents are adapting their negotiation plans according to an evolved model of the other agents in the system.

An argumentation based negotiation protocol is proposed in [Kakas A. and Moraitis P., 2006], in which offers of the negotiating parties are linked to different arguments they can build, according to their individual negotiation strategy. This protocol is able to take into account the different roles of agents and the context of interaction, where the strength of the arguments supporting an offer can depend on these factors. The agents can adapt their negotiation strategies and offers, as their environment changes, in particular during the course of the negotiation, as they exchange information. In addition, using abduction alongside with argumentation, agents can find negotiating conditions to support an argument for an offer, thus extending the negotiation object in order to help finding an agreement. To illustrate further the advantages of the approach, the negotiation strategies are extended with another negotiation mechanism, that is bargaining with multiple parties.

4.5. Negotiation Systems with Ontologies

In a multi-agent system, presented in [Dong H. et al., 2008], a business may start by two agents that have the desire of trading, and negotiation is an inevitable procedure for building the business relationship. In the context of e-business, agents

that represent humans have the potential ability to start automated negotiation activities.

Traditional negotiation research focuses on providing approaches for generating negotiation strategies and protocols to enhance the ability of agents. However, there are some issues in this field – the inadaptability of agents to evolving negotiation protocols, and agents' negotiation term ambiguity.

Ontology is a semantic web technology for defining domain knowledge and solving semantic ambiguity, which can be extended into the field of automated negotiation research. A survey of the existing negotiation ontologies is performed, and the current status of the negotiation ontology research is explored.

Semantic web technology to automated negotiation is applied in [Tamma V. et al., 2005]. In this approach, agents could negotiate in any type of marketplace, irrespective of the negotiation mechanism. Protocols are described in terms of a shared ontology, which is an explicit and declarative representation of the negotiation protocol. The ontology is used to change agents' strategies to a certain protocol employed.

An agent entering an interaction should acquire the negotiation protocol, which controls the interaction, from the marketplace itself, through an advertisement of the type of protocol used. In order to allow interoperability, the protocol is defined like a shared ontology of negotiation, which gives the basic vocabulary that agents must share, in order to discuss the participation terms in the negotiation.

Regarding the use of the ontology as input for changing the agent strategy to a certain type of protocol, it is described the complexity of the protocol defined by a given mechanism and recommends a potential solution technique. For some mechanisms, it is possible to find an optimal strategy using computational techniques. For more complex mechanisms, this approach can be used to recommend a strategy, based on a class of learning algorithms.

4.6. Negotiation Systems in Auctions

The multi-agent paradigm and its application to multi-item auctions is discussed in [Benameur H. et al., 2002]. It is proposed a formal model for auction based automatic negotiations. This model is implemented using multi-agent systems and is tested and evaluated with simulation experiments.

Multi-item auctions have addressed the combinatorial issue that allows bids on combinations of items, as opposed to only single items. These approaches suppose two simplifying conditions: the quantity of items to sell is fixed, as well as the quantities requested by the buyers. These two hypotheses do not meet the requirements of many situations, where auctions are used. In some auctions, it is desirable that the available quantity is not fixed. In this way, quantities can change during the auction, as it is for example the case for stock values. A model based on

an English auction is presented, with multiple items with private valuations and variable quantities requested.

4.7. Heuristic Negotiation Systems

The design and implementation of an environment for automated negotiations, offering support for the use of various existing bargaining models, together with their respective negotiation strategies, is presented in [Silva A. et al., 2007]. It is possible to integrate different models, approximating the automated negotiations to the way the real world works.

An open architecture for the negotiation environment is proposed, where the number of buyers and sellers, or the offer and demand for services or products, can be changed during execution time. In a flexible way, the negotiator agent can increase the set of negotiation strategies and also the number of business domains. These characteristics were implemented in the proposed environment with ontology concepts and production rules.

The use of ontology enabled the negotiator agents to be implemented, a priori, for any type of business domain. The fact is that the environment supplies a protocol for the agents to interact on the same ontology, allowing an agent to be detached from the various business domains it may be possible to negotiate. The individuality of each negotiator agent is therefore tied to the rules it possesses.

Price negotiations are one of important aspects of e-commerce transactions [Badica C. et al., 2007]. A rule-based implementation of automated price negotiations, used in a multi-agent system that models an e-commerce environment, is presented. A brief description of the conceptual architecture of the system and a simplified scenario that involves multiple buyer agents participating in multiple English auctions performed in parallel are described.

Dominant-strategy mechanisms in allocation domains, where agents have one-dimensional types and quasilinear utilities, are studied [Naroditskiy V. et al., 2013]. Considering as input an allocation function, an algorithmic technique for finding optimal payments is presented, for a class of mechanism design problems. Optimality of payment functions is linked to a geometric condition. When the condition is true, an optimal payment function that is piecewise linear in agent types is described.

Mechanism design problems that have no objective functions, but seek payments fulfilling a combination of constraints, are reduced at solving a system of linear inequalities. These reductions give solutions of mechanism design problems that are otherwise difficult to solve.

In Table 4.1 are presented the types of research on automated negotiation and the authors.

Table 4.1. Types of Research and Authors

Types of Research	Authors
Negotiation using Decision and Game Theory	Kraus S., 1997 Sierra C., Faratin P., Jennings N.R., 1997 Kraus S., 2001 Rahwan I., Kowalczyk R., Pham H.H., 2002 Osborne M.J., 2004 Narayanan V., Jennings N.R., 2005 Choi H.R., Kim H.S., Hong S.G., Park Y.J., Park Y.S., Kang M.H., 2005 Brzostowski J., Kowalczyk R., 2006 Fatima S., Wooldridge M., Jennings N.R., 2007 Fatima S., Wooldridge M., Jennings N.R., 2010
Learning Algorithms used in Negotiation	Chen Y., Peng Y., Finin T., Labrou Y., Cost S., 1999 Soh L.-K., Li X., 2004 Lau R.Y.K., 2005 Crawford E., Veloso M., 2005 Lau R.Y.K., Tang M., Wong O., Milliner S.W., Chen Y-P.P., 2006 Florea A.M., Kalisz E., 2008
Negotiation using Argumentation	Florea A.M., 2002 Carabelea C., 2002 Kakas A., Moraitis P., 2006 Rahwan I., Ramchurn S.D., Jennings N.R., McBurney P., Parsons S., Sonenberg L., 2002
Negotiation Ontologies	Dong H., Hussain F.K., Chang E., 2008
Negotiation in Auctions	Benameur H., Chaib-draa B., Kropf P., 2002

Table 4.2 describes the main characteristics of each negotiation model developed in the researches presented in this chapter.

Table 4.2. Main Characteristics for Negotiation Models

Research Authors	Name/ Language	Adaptation	Learn	Single/ Multi
Sierra C. et al., 1997	CLIPS and PROLOG	Fuzzy Control Techniques	No	Multi
Bratman M.E., 1999	Theoretical Framework	Heuristic Strategies	Reinforcement Learning	Single
Chen Y. et al., 1999	Supply Chain Management System	Colored Petri Nets	Distributed Constraint Satisfaction Problem	Multi

Benameur H. et al., 2002	Java	Finite State Machines	No	Multi
Carabelea C., 2002	Jade	Argumentation Strategies	Reinforcement Learning	Multi
Florea A.M., 2002	Java	Argumentation Strategies	Reinforcement Learning	Multi
Rahwan I. et al., 2002	Intelligent Trading Agency (ITA)	Constraint-based Techniques	No	Multi
Soh L.K. and Li X., 2004	Java	Multi-Phase Coalition Formation	Form Coalitions, Perform Global Tasks	Multi
Choi H.R. et al., 2005	Server Sales Agent System / Java	Scheduling Agent / Price and Due Date Agent	No	Multi
Crawford E. and Veloso M., 2005	Java	Playbook Approach	Experts Problem / Regret	Multi
Narayanan V. and Jennings N.R., 2005	Java	Markov Decision Process	Non-Stationary Value Iteration Algorithm	Multi
Tamma V. et al., 2005	DAML + OIL / Protege	Semantic Web Technologies	Q-learning	Multi
Brzostowski J. and Kowalczyk R., 2006	Java	Non-linear Regression	Predictive Decision Making	Multi
Kakas A. and Moraitis P., 2006	Java	Argumentation Strategies	Abduction	Multi
Lau R.Y.K. et al., 2006	Web Service Description Language (WSDL)	Genetic Algorithms	Evolutionary Mechanisms	Multi
Badica C. et al., 2007	Jade and Jess	Multiple English Auctions Performed in Parallel	No	Multi
Fatima S.S. et al., 2007	Theoretical Framework	Equilibrium Strategies	Approximately Optimal	Multi
Silva A. et al., 2007	Java	Neural Networks	Q-learning	Multi
Dong H. et al., 2008	Protege and Jade	Semantic Web Technologies	No	Multi
Florea A.M. and Kalisz E., 2008	Java	Heuristic Strategies	Reinforcement Learning	Multi
Fatima S.S. et al., 2010	Theoretical Framework	Time Constraints	Polynomial Time Methods	Multi
Naroditskiy V. et al., 2013	Theoretical Framework	Dominant Strategy Mechanisms	Optimal Payment Function	Single

Part II. An Adaptive Negotiation Multi-Agent System

Chapter 5. Automated Negotiation using Profiles and Clustering of Agents

This chapter presents a model of self-interested agents acting in an open environment, which capture the most relevant elements of agents' behavior related to negotiation with other agents. The agents' behavior is mainly motivated by the gain they may obtain while fulfilling their goals and negotiating, but their behavior can change during negotiation, according to previous interactions with other agents in the system.

5.1. Multi-Agent System Description

A series of negotiation strategies and studies already exist in the literature, for example [Jennings N.R. et al., 2001; Ito T. et al., 2007; Jonker C. et al., 2007; Lin R. et al., 2008; Lin R. et al., 2009 a], in which the agents are able to choose different strategies. The automated negotiation process usually takes place in open environments. These environments don't have a way to control the agents' behavior, and it is also possible to have humans in these environments, whose behavior is unpredictable [Florea A.M. and Radu S., 2007; Lin R. et al., 2008]. Some negotiation strategies are based on agent profiles, which can define statically or develop dynamically agents' preferences. There are advantages for creating agents' negotiation profiles. This allows the agents to build good strategies. Using these profiles, agents obtain better results than those in case of fixed negotiation strategies, that is increase the agents' gain from negotiation [Florea A.M. and Kalisz E., 2007; Radu S. and Florea A.M., 2012].

Changing the behavior of the agents may refer to either the use of different negotiation strategies or to concessions made for other agents, with which they have successfully negotiated in the past. Sometimes, the agents are also motivated by the necessity to cooperate with other agents for achieving their goals. The key element in the agent behavior is their capability to develop a set of negotiation profiles. These profiles help the agents to conduct their negotiation [Radu S. et al., 2013 a]. Different approaches to the development of these profiles are presented in this chapter.

The set of negotiation profiles the agents are able to evolve consists of:

(1) the **preference profile**, defining the agent negotiation strategy;

(2) the **partner cooperation profile**, which takes into account the agent interaction with the other agents in the system;

(3) the **group-of-partners' negotiation profile**, which clusters the profiles of several agents.

The first two profiles characterize individuals, while in a group negotiation profile, several agent profiles are clustered according to commonly discovered features.

The outcome of negotiation is evaluated for different strategies, encoded in the preference profile. In the system there are different agents and a **facilitator**, which can be used by the agents, either to register or, to register and facilitate negotiation.

The three types of negotiation profiles are described and discussed, as well as the **uninformed and informed negotiation strategies**. Negotiation strategies are implemented in the form of **production rules**. In this approach, **preference coefficients** can be assigned to these rules and dynamically modified, according to the negotiation situation.

The proposed model of negotiation is tested in the framework of a multi-agent system, for different business models. The results are presented in detail in Chapters 8 and 9.

5.2. Overview of the Approach using Profiles and Clustering of Agents

A model of heuristic negotiation between self-interested agents is described in [Florea A.M. and Kalisz E., 2007]. The system allows bargaining over multiple issues of the negotiation object, comprises different types of negotiation primitives, including argument based ones, and a set of rules to conduct negotiation. In order to negotiate strategically and to adapt bargaining to different partners, the agents use **rewards** associated to negotiation objects and the notion of **regret** to compare the achieved outcomes with the best possible results that could have been obtained both in a particular negotiation and in selecting the partner agent.

The decision process during the negotiation is modeled as an adversarial bandit problem with partial information and uses the computed probabilities of negotiation rules to select the best rule to be used at a certain moment during bargaining. Rewards were defined depending on the attributes of the negotiation object at a given negotiation round or, in case adaptation of partner selection is sought, depending on the negotiation object with which the negotiation is concluded. Moreover, it is shown how the problem can be modeled if not all rules can be selected at a given decision point (equivalent with not all experts being available for consultation).

The Q-learning algorithm is applied to analyze and learn customer behaviors and then recommend appropriate products in [Srivihok A. and Sukonmanee P., 2005]. As compared to the current approach, the user profile is not used for negotiation, but to personalise the information to the customer interests. There are used weighting features to recommend products to the user. In this chapter, weights to represent the preference coefficients are employed.

A software framework for negotiation, in which the bargaining mechanism is represented by a set of rules, as in the current chapter, is proposed in [Bartolini C. et

al., 2005]. The rules are organized in a taxonomy and can be used in conjunction with a simple interaction protocol. The negotiation language is based on OWL-Lite. Although the rules allow flexible definition of several negotiation strategies, there are no negotiation profiles and the possibility to modify the negotiation, according to these profiles, as in the current chapter.

An implementation of automated negotiations in an e-commerce modeling multi-agent system is described in [Badica C. et al., 2006 b]. A specific set of rules is used for enforcing negotiation mechanisms. An experiment involving multiple English auctions performed in parallel is discussed.

A system for automated agent negotiation, based on a formal and executable approach to capture the behavior of parties involved in a negotiation is shown in [Skylogiannis T. et al., 2007]. The negotiation strategies are expressed in a declarative rules language, defeasible logic, and are applied using the implemented system Dr-Device.

The system GENIUS (General Environment for Negotiation with Intelligent multi-purpose Usage Simulation) is presented in [Lin R. et al., 2008]. It supports the design of different strategies for agent negotiation, and the evaluation of these strategies in a simulated environment. The system allows the negotiation between automated agents, but also between agents and humans. The designer of a strategy can select from a repository a negotiation domain and a preference profile for the agent. Both are represented in a tree-like structure, which enables to specify priorities related to outcomes of negotiation. As compared to this system, in the current chapter, there are several negotiation profiles, which are evolved during interactions, and the negotiation domain is specified by the agent rules.

5.3. Framework for Automated Negotiation

The negotiation agent behavior is defined in a framework of a multi-agent system in an open environment. The system includes the facilitator, which can be used or not by the agents during negotiation, depending on the preference profile. Because the environment is open, other agents can enter or exit the system dynamically, as presented in Figure 5.1. When entering the system, the agent has to register with the facilitator, and the facilitator will inform other agents about the new agent arrival.

There are two interaction possibilities, each one with its advantages and disadvantages. When an agent wants to negotiate, it either sends a broadcast message to all the agents in the system, or sends a single message to the facilitator, asking it to find other agents appropriate to negotiate with.

In the first approach, in which the agent sends messages to all the other agents in the system, the agent waits for answers from possible partners. Some of the agents will answer with proposals and others will not answer at all. The advantage is that the

facilitator has fewer messages to process, but the disadvantage is that there are too many messages sent over the network, which can slow down the transaction.

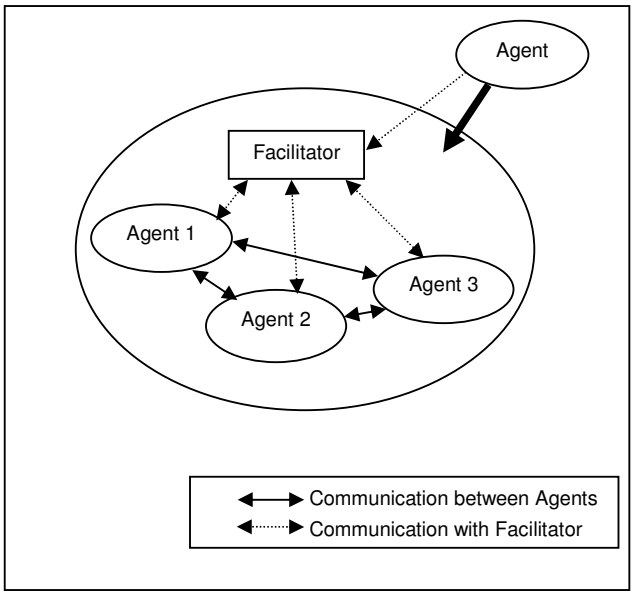


Figure 5.1. The Multi-Agent System Open Environment

In the second approach, the agent will send to the facilitator two types of messages: a message presenting its abilities, and a message asking help for a negotiation, for instance, to buy or sell an item. It is the facilitator which will query the agents with the required abilities, and will inform the agent about the outcome. The advantage of this approach is that not all the agents are queried, but the disadvantage is that the facilitator has a lot of messages to process and can become a bottleneck in the system.

The negotiation between agents is a single-issue negotiation, where the price of the products is bargained. The type of negotiation implemented is a heuristic negotiation, in which the agent computes the gain, as compared to its private value for an object. The agents negotiate following the **Iterated Contract Net** protocol. Each agent takes into account a private value for any item (product or service) to trade (sell or buy). Also, there is a deadline for the number of negotiation rounds. In a buying negotiation, an agent will look for a lower value than its private one, while in a selling negotiation, its main goal is to obtain more than the item's private value.

The agent behavior is mainly motivated by the gain, but also, depending on a specific context, by the desire to achieve cooperation with other agents. For example, as a general rule, the agent will not accept a price lower than its private value. However, when the agent wants to cooperate, it can accept a lower price. The agent behavior is set up by the negotiation strategy.

The negotiation model proposed in this chapter can be easily extended to multi-issue negotiation, as described in Chapter 6. In this case, the agent will trade the

negotiation object (NO). A negotiation object is the range of issues over which agreements must be reached, as defined in [Jennings N.R. et al., 2001]. The object of a negotiation may be an action which the negotiator agent A asks another agent B to perform for it, a service that agent A asks to B, or, alternatively, an offer of a service agent A is willing to perform for B, provided B agrees to the conditions of A. The agent A may have a plan to achieve one of its desires, but may be unable to carry out some of the necessary actions, therefore it will ask B to execute these actions for it, if it believes the actions belongs to B's abilities.

The agents are mainly designed according to the BDI model, and have a set of goals, selected from the set of desires. In order to achieve their goals, they develop plans, as a sequence of actions to be performed, or they provide services or ask for services from the agents, such as buying or selling objects. Some actions can not be executed by the agent itself, and they also become, together with the services, objects to be negotiated by the agents. To unify these two cases, the agents are using negotiation objects, which can wrap up one or several negotiation attributes.

Each agent has an associated set of rules, divided in two: **behavior rules**, which implement the way the agent fulfills the goals assigned to it, and **negotiation rules**, which describe the negotiation strategy. In Figure 5.2 is presented the structure of a BDI negotiating agent. Agents based on models different from BDI may enter the system, providing that they use the same negotiation protocol. During negotiation and interaction with other agent, a BDI agent develops a set of negotiation profiles.

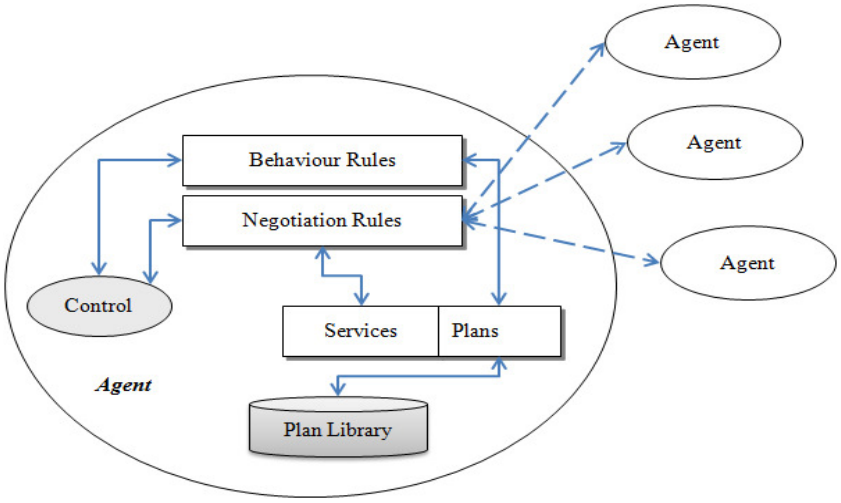


Figure 5.2. BDI Negotiation Agent Structure

The agents have different reasoning capabilities, designed to conduct successful negotiation and the fulfillment of agents' goals. During negotiation, the agents gather

information about the partner agents, and store it in the associated cooperation profiles [Radu Ș. et al., 2013 a].

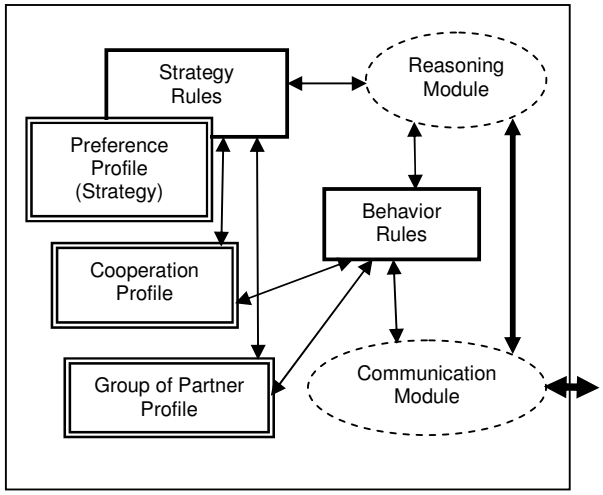


Figure 5.3. The Agent Main Components

Each agent is endowed with a set of negotiation profiles, outlined in Figure 5.3:

- **the preference profile**, which specifies the agent negotiation strategy;

- **the partner cooperation profile**, which keeps track of the agent interactions with the other agents in the system;

- **the group-of-partners' negotiation profile**, which deals with a group of negotiation partners.

The partner cooperation profile describes the preferences regarding the agents with which an agent prefers to cooperate. The partner cooperation profile of an agent is implemented as a structure which evolves dynamically. For each agent encountered during the negotiation process, knowledge about the outcome of different negotiations is stored in this structure.

The partner cooperation profile is a characteristic of each agent and is stored as a matrix. Each line contains the agent name and a set of associated attributes, upon which the agent dynamically changes its preferences. The cooperation profile is updated during the negotiation process, at the end of each negotiation. Table 5.1 presents an example of a partner cooperation profile for an agent.

An entry in the matrix, defining the partner cooperation profile for an agent, has the fields presented below, representing the cooperation attributes.

1. The first field represents the **name of the partner agent**, stored as a string of characters.

2. The second field contains **how many times the agent negotiated with its partner**, and is represented as a natural number.

3. The third field stores **the number of successful negotiations**, as an integer number.

Table 5.1. Partner Cooperation Profile of an Agent

1. Partner Agent Name	Agent 1	Agent 2
2. No. of Negotiations	7	5
3. Successful Negotiations	4	3
4. Total Gain	3	2
5. Gain Ratio	80	60
6. Negotiation Rounds	6	4
7. Interesting Degree	3	2
8. Partner Classification	very cooperative	cooperative

4. The fourth field points out the connection between the outcome of negotiation and the private value of the agent for the negotiation object, showing the **total gain** obtained. This is stored as a positive or negative integer number.

Each agent has two prices, the **minimum price** (MinPrice) and the **maximum price** (MaxPrice), between which it accepts offers. The **buyer** computes its **gain** as the difference between the maximum price it is willing to pay and the price of the current offer. The **seller** computes its **gain** as the difference between the price of the current offer and the minumum price it is willing to accept.

Based on heuristic criteria, an agent can accept a price greater than its private value in order to buy a product, or can sell a product at a price lower than its private value. These decisions are based on the information stored in the preference profile of the agent and are specified by the strategy rules.

5. The fifth field shows the **gain percentage**, namely how much is the gain obtained while negotiating with the partners, as a percentage of the agent total gain.

6. The sixth field is the **number of negotiation rounds**, during the last negotiation, and is an integer number.

7. The seventh field tries to capture the agents' beliefs about the partner abilities and/or credentials. This attribute is called the **interesting degree of the partner**, which is quantified as: **very interesting**, **interesting**, **moderately interesting,** and **not interesting** (4…1). For example, if the partner has an ability to perform a task, which is lacking to the agent, then the partner is interesting or very interesting to the agent. Moreover, if the negotiation is successfully concluded in a small number of steps, and the gain is positive, then the partner is very interesting.

8. The eighth field contains the **classification of the partner agent**, which represents the current agent belief about the cooperation potential of the partner. The partners are classified in **six cooperation classes**: **highly cooperative**, **very cooperative**, **cooperative**, **slightly cooperative**, **non cooperative**, and **unknown**.

The attributes from **1** to **7** described above are updated by the agent after each negotiation with a specific agent. The last attribute (**8**) will be filled in by a more elaborate process, to be described further on.

The representation of the agents' cooperation profiles as a set of attributes is able to characterize the cooperation potential of a partner, with a high degree of

granularity. It is necessary to describe the cooperation potential of a partner agent in a broader sense. To this aim, it is proposed the clustering of partners into cooperation classes, specified in field **8** [Radu S. et al., 2013 b]. This classification is achieved by using the **C4.5 learning algorithm** [Quinlan J.R., 1993], in which the fields from **2** to **7** are used as classification attributes and field **8** represents the class. There are two methods to generate training examples for the learning algorithm. In the first approach, the system run and collect gathered information. In the second approach, a training set of virtual agents is generated, with which an agent has virtual negotiations. After these negotiations are performed, the matrix is filled with the results, according to the fields described before.

The information stored in this way in the matrix will allow the classification of real negotiations. The classification is improved, as more negotiations take place and the matrix is filled with the correct real values. After that, the C4.5 algorithm will classify correctly the negotiation instances.

5.4. Agents Classification and Strategies

The characterization of the cooperation potential of a partner agent is done by classifying the partners into cooperation classes, which divides the cooperation ability of the partner into six classes. The classification is done using the C4.5 learning algorithm, in which a decision tree is a classifier for the cooperation degree, expressed as a recursive partition of the instance space. In the decision tree, the attributes are represented by the first seven fields of the partner cooperation profile and the class by the partner classification field.

Decision trees are capable of handling datasets that may have errors. Also, they can handle datasets that may have missing values, like the gain value in the current case.

A tree is either a leaf node labeled with a cooperation class, or a structure containing a test for an attribute, like the number of successful negotiations, linked to two or more nodes (or sub-trees). So, to classify some instance for the cooperation potential, first it is obtained its attribute-vector, and then this vector is applied to the tree. The tests are performed with these attributes, like the number of successful negotiations, the gain, the gain percent, reaching one or other leaf, to complete the classification process.

Through a top-down decision tree and a heuristic selection criterion, the process chooses the best test to split the data, creating a branch.

In order to build the classification tree, a set of training instances is necessary, with associated attributes, and a corresponding class. There are two possibilities to obtain the set of training instances. The first possibility is to let the system run and collect information, based on agent interaction. The tree is built after a number *n* of negotiations, and then rebuilt at successive times, *n+1*, *n+2*, increasing thus the accuracy of the classification.

The other possibility is to create a set of training instances, and run the C4.5 algorithm on this set. An initial classification tree will be thus obtained, which may be later on rebuild and refined, after the actual negotiation will take place.

The C4.5 algorithm should work with inadequate attributes and should obtain correct results. Also, it should decide if testing some supplementary attributes will increase or not the predictive accuracy of the decision tree. Considering an attribute *A* with random values, choosing this attribute will give a high informational gain. The benefit should be greater than a given threshold, in order to eliminate the non relevant attributes.

There are cases when the classification can't be done. This happens when a leaf is obtained, in which not all the objects belong to the same cooperation class. In this case, the notion of membership to a cooperation class with a certain probability is used or the leaf is labeled with the cooperation class with the largest number of instances. If the classification ends in a leaf with an equal number of instances from two cooperation classes, the decision about the correct classification in a cooperation class is done randomly.

When there are missing attribute values in the training set, for instance, the interesting degree of a partner, it is assigned the value of that attribute, which appear most often. A different approach for missing attributes is to assign values distributed over the values of the attribute *A* proportionally to the relative frequency of these values in the set of objects.

Figure 5.4 presents a part of the decision tree used for classifying the instances.

While interacting with the partners, the agents are classified in the right class. At the end of negotiation, an agent can change its cooperation class. There is a tradeoff between how often the agent cooperation class is updated, which may be time consuming, and the accuracy of the classification.

The group-of-partners' negotiation profile is defined by grouping into classes the partner agents, with which the agent interacted in the system. For each of the six values of the cooperation classes (highly cooperative, very cooperative, cooperative, slightly cooperative, non cooperative and unknown), the group-of-partners' negotiation profile contains the list of all agents that belong to a given class. The agents, for which no class was found out yet, belong to the unknown class.

The preference profile of an agent describes its strategy. This is represented as a set of negotiation rules. These can be heuristic negotiation rules, which refer to partner cooperation profiles, or rules which encode certain negotiation strategies.

The negotiation strategy of an agent may show how much and how quickly the prices are decreased and if the price is lower than the agent's private value. An agent can use a tradeoff in negotiation. In the case of cooperative agents, an agent can gain more once, and then can sell cheaper. This is a kind of global evaluation on previous deals.

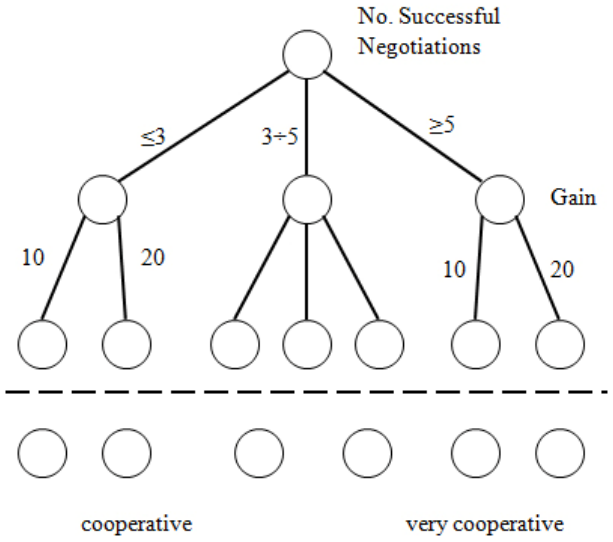


Figure 5.4. Classification using the Partner Cooperation Profile

The negotiation strategy is implemented in the form of production rules, each agent keeping a history of its interactions. If in a given situation, several rules are eligible, then the negotiation strategy decides which rule, from the conflict set, to be applied. A possible approach to solve the conflicts is to assign priorities between rules. The solution is to apply the rule with the highest priority. Using a feedback, it is possible to apply the same rule or another rule.

The rule's priorities are dynamically modified, and the preference coefficients are not built-in, they are dynamically changed, according to the negotiation situation.

Each rule has an associated preference coefficient, which indicates, in case two or more rules apply in a given situation, which rule should be preferred. In this way, the strategy can be explicitly set up when designing the system, and can be changed from one use of the system to another, simply by changing these coefficients. Another use of the coefficients is that they can be dynamically modified by a **reinforcement learning algorithm**, in a similar manner to a learning classifier system [Butz M.V., 2010]. Learning classifier systems are a machine learning technique that may be categorized between symbolic production systems and sub-symbolic connectionist systems.

It is possible to have more than one strategy in the system, for instance, at each negotiation step, the price is decreased by 1, or is decreased by 3. Another strategy rule tells what happens when the price increases with 10 % above the private value, if the offer is instantly accepted or not.

In order to show an example of strategy rules, the following Contract Net protocol is considered:

1) ***cfp(A, X, NO, P)*** is the communication primitive, which represents a call for proposals from agent *A* to all the acquaintances *X*, regarding a negotiation object *NO*, with an associated cost *P*

2) ***propose(X, A, NO, P, Step)*** is the communication primitive, which represents the response of agent *X* to the *cfp*, with the negotiation object *NO*, price *P* and negotiation step *Step*

3) ***accept(X, NO)*** indicates the acceptance of a proposal issued by *X*, for the *NO*

4) ***reject(X, NO)*** indicates the rejection of a proposal issued by *X*, for the *NO*

5) ***counterpropose(A, X, NO_1, P_1, Step)*** defines a communication primitive, which represents the counterproposal of agent *A* to the proposal of agent *X*, with the negotiation object NO_1, price P_1 and negotiation step *Step*

A Prolog-like language [Florea A.M. et al., 2007] is considered and also a set of predicates, defined as follows:

- *propose(X, A, NO, P, Step)* defines a predicate, which is true when agent *A* receives the response *(2)* of agent *X* to the *cfp*, with the negotiation object *NO*, price *P* and negotiation step *Step*
- *accept(X, NO)* defines a predicate which, when true, triggers an acceptance message *(3)* of a proposal issued by *X*, for the *NO*
- *reject(X, NO)* defines a predicate which, when true, triggers a rejection message of a proposal issued by *X*, for the *NO*
- *counterpropose(A, X, NO_1, P_1, Step)* defines a predicate which, when true, represents the counterproposal *(5)* of agent *A* to the proposal of agent *X*, with the negotiation object NO_1, price P_1 and negotiation step *Step*
- *tp(Ag_Name, Atr_Name, Value)* is a predicate which selects from the partner cooperation profile, for a given agent name (*Ag_Name*), the value (*Value*) of the attribute (*Atr_Name*) in the associated field.

Considering the above described predicates, some examples of strategy rules of an agent are given in Table 5.2.

During negotiation steps, the C4.5 learning algorithm can classify the partner in another cooperation class, if the criteria used as attribute in the algorithm are changed.

So, instead of having a negotiation strategy which applies all the rules that match at a certain moment, the classification with the C4.5 learning algorithm decreases the number of eligible rules.

At a higher degree of granularity, instead of writing negotiation rules for a partner cooperation profile, we can write negotiation rules for a group-of-partners' negotiation profile. Moreover, the group-of-partners' negotiation profile can be used in other ways to tailor the agent behavior.

Table 5.2. Example of Strategy Rules

Id	Rule
r_1	*propose(John, Tom, House, 1000, S),* $tp(John, No\_Successful\_Negotiations, v_1)$, $tp(John, No\_Negotiations\_with\_Partner, v_2)$, $v_1 > v_2 - 2$, $tp(John, Interesting\_Degree\_of\_Partner, v_3)$, $v_3 > 3$, $tp(John, Gain\_Percent, v_4)$, $v_4 > 20 \rightarrow$ *accept(John, House)* PC_1
r_2	*propose(John, Tom, House, 1000, S),* $tp(John, No\_Successful\_Negotiations, v_1)$, $v_1 > 5$, $tp(John, Gain, v_2)$, $v_2 > 100$, $tp(John, Interesting\_Degree\_of\_Partner, v_3)$, $v_3 = 4 \rightarrow$ *accept(John, House)* PC_2
r_3	*propose(John, Tom, House, 1000, S),* $tp(John, No\_Successful\_Negotiations, v_1)$, $tp(John, No\_Negotiations\_with\_Partner, v_2)$, $v_1 < 0.5 * v_2$, $tp(John, Interesting\_Degree\_of\_Partner, v_3)$, $v_3 < 2 \rightarrow$ *reject(John, House)* PC_3
r_4	*propose(John, Tom, House, 1000, S),* $tp(John, No\_Successful\_Negotiations, v_1)$, $tp(John, No\_Negotiations\_with\_Partner, v_2)$, $v_1 < 0.5 * v_2$, $tp(John, Interesting\_Degree\_of\_Partner, v_3)$, $v_3 > 3 \rightarrow$ *counterpropose(Tom, John, Car, 500,* S_1*)* PC_4
r_5	*propose(John, Tom, House, 1000, 1),* $tp(John, Classification, v_1)$, v_1 = *'highly cooperative'* $\rightarrow$ *accept(John, House)* PC_5
r_6	*propose(John, Tom, House, 1000, S),* $tp(John, Classification, v_1)$, v_1 = *'unknown', Price > private_value(House)* $\rightarrow$ *reject(John, House)* PC_6
r_7	*propose(John, Tom, House, 5000, 1),* $tp(John, Classification, v1)$, *v1 = 'unknown', private_value(House) = p,* $p < 5000$, $tp(John, Gain, v2)$, *v2 = 5000 - p* $\rightarrow$ *accept(John, House),* $tp(John, No\_Negotiations\_with\_Partner, 1)$, $tp(John, No\_Successful\_Negotiations, 1)$ PC_7

5.5. Preference Coefficients Determination

The preference profile implements the negotiation strategy using rules, with the associated preference coefficient [Radu S. et al., 2013 b].

There are two types of rules used in the strategy. The first category of rules is used when the negotiation begins and is called **uninformed strategy rules**. The second category is employed when there is enough information about the partner and is called **informed strategy rules**. When negotiation begins, uninformed strategy rules can be applied. According to the success or failure of the negotiation, the preference coefficients are changed accordingly.

In order to update the preference coefficients, it is established a formula, which makes the connection between the preference coefficients and how much the agent gains using the rules associated to these coefficients. It is supposed that

all the rules applied during negotiation have an equal contribution to gain or to loss. For a certain agent acting in different negotiations, if there are *n* possible rules to be applied, a priority between the rules will be established, according to the preference coefficients. Suppose that N_R represents how many times the rule *R* was applied in a negotiation and *M* is the number of negotiation steps. Denote by α the ratio between N_R and *M*.

$$\alpha = N_R / M \tag{5.1}$$

The preference coefficient is updated according to the situation, if the negotiation ends with a deal or not. If a deal is reached at the end of the negotiation, then the preference coefficient is updated by the formula (5.2):

$$PC_{new} = \begin{cases} (1+\alpha) * PC_{old} & PC_{old} \in [0 \ldots 1] \\ (1+\alpha) * PC_{old} \div 2 & PC_{old} > 1 \end{cases} \tag{5.2}$$

If a negotiation is ending without a deal, then the preference coefficient is updated by the formula (5.3):

$$PC_{new} = (1-\alpha) * PC_{old} \tag{5.3}$$

Some predefined values for the preference coefficients are put in the beginning. Also, there is a mechanism used to adjust the coefficients and to realize a combination between the initial preferences of the user for the rules and the change in time of the coefficients, according to the result of the negotiation.

A second approach used to update the preference coefficients is a reinforcement learning algorithm. In reinforcement learning, agents revise their strategies based on observed failure or success. In **Q-learning** [Tesauro G. and Kephart J., 2002], a reward function provides feedback on actions taken in order to estimate a ranking of state-action pairs.

To apply the Q-learning algorithm in this situation, it is considered that each preference coefficient is indexed on a state and an action, for taking into account the preference coefficients. The actions from the Q-learning algorithm are represented by the rules applied by an agent when negotiating. The states from the Q-learning algorithm represent now the internal states of the agents. For each tuple (*s, a*), where *s* is the internal state of the agent and *a* represents the rule applied during negotiation, the preference coefficients are updated using a formula, in a similar manner to the Q-learning algorithm:

$$PC(s,a) \leftarrow PC(s,a) + \alpha[r(s) + \gamma \max_{a'} PC(s',a') - PC(s,a)] \tag{5.4}$$

where:

$\max_{a'} PC(s',a')$ is the expected preference coefficient of the next internal state of the agent *s'*, when applying the action *a'*;

α is the learning rate representing the impact of the update value;

r(s) is the immediate reward for the internal state of the agent *s*.

The immediate reward in this case is the gain obtained during negotiation. The factor γ specifies how much the values of the preference coefficients are

discounted at each stage.

The internal state of the agent is characterized by several parameters: the partner agent, the negotiation object, the utility of the current offer, the number of negotiation parameters, the number of negotiation rounds with the partner agent, how much the agent gained in a previous negotiation.

To adequately update the preference coefficients using the Q-learning algorithm, the agent must encounter repeatedly the same pair *(s, a)*, therefore the negotiation must be performed several times with the same agent and for the same object.

It is important to consider that the matrix *PC(s, a)* is huge, because it is possible to have many internal states of the agent, for each possible combination of the parameters for its internal states. For instance, if there are ten negotiation rules possible to be applied in a certain state, then the matrix will have ten columns and the lines represent different combinations of state parameters. One line can represents the following internal state: the name of the partner agent, the negotiation round and the gain obtained in a previous negotiation. It is clear that there are too many internal states, for each possible combination of state parameters and the number of lines of the matrix increase exponentially, as the number of state parameters grows. That's why the learning algorithm used for updating the preference coefficients will converge in a very long time.

An idea to reduce the matrix dimensions of *PC(s, a)* and to improve the convergence time of the learning algorithm is to group the internal states of the agent, according to the **k-means algorithm** [Pena J.M. et al., 1999]. Specifically, clusters of internal states are created and the number of lines of the matrix will significantly decrease, because each line will represent a cluster of states. In this way, the learning algorithm will converge in a reasonable time and the values of the preference coefficients are updated at the end of the learning algorithm application. An example of clustering for the internal states of an agent, according to the parameters of its internal state, is presented in Figure 5.5 [Radu Ș. et al., 2013 b].

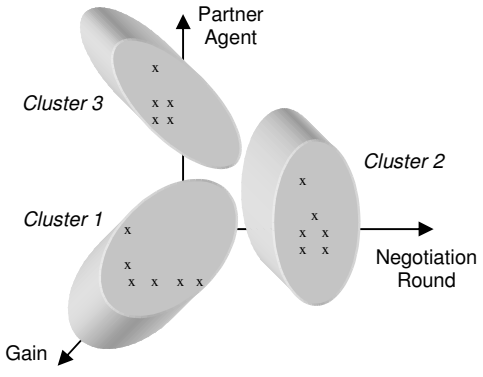


Figure 5.5. Clustering the States of the Agent

Each state representing the output of a negotiation is a point in a multidimensional space. The algorithm classifies the data set through a certain number of *k* clusters. The idea is to randomly define *k* centroids, one for each cluster. It is better to put the centroids as much as possible far away from each other. A better way to initialize the centroids is to use the **k-means++ algorithm** [Arthur D. and Vassilvitskii S., 2007], in which the first centroid is randomly choosen from the initial data set. Then, each centroid is picked out from the remained objects, with a probability:

$$\frac{D(x_i)^2}{\sum_{x \in X} D(x)^2} \tag{5.5}$$

for each object $x_i \in X$, where $D(x)$ is the smallest distance between the point *x* and a centroid already chosen.

The next step is to take each point which belongs to a certain data set and associate it to the closest centroid. When no point is pending, the first step is completed and an early clustering is done. At this point, it is necessary to compute again *k* new centroids as centers of the clusters resulting from the previous step. After these *k* new centroids are computed, a new binding is done between the same data set points and the nearest new centroid. A loop has been created. As a result of this loop, it is observed that the *k* centroids change their location step by step, until no more changes could be done. In other words, centroids do not move any more.

Using the k-means algorithm, it is possible to group the internal states of an agent into clusters. If the preference coefficients should be updated and improved, the same negotiation must be performed in the same conditions several times. The clusters decrease significantly the number of negotiations performed. Therefore, in order to update the coefficients, a smaller number of negotiations are performed. The time in which the preference coefficients are updated is reduced and the negotiation time is decreased. The coefficients are adjusted in a shorter time, when using clusters, than in the case of individual negotiations.

5.6. Conclusion

In this chapter, it is presented a model of negotiating agents that aimes to combine the agents beliefs about the other agents in the system, with the possibility to explicitly represent and modify the negotiation strategy, expressed in a set of rules. In order to achieve this, there are defined three negotiation profiles: the preference profile, the partner cooperation profile and the group-of-partners' negotiation profile. The last two, partner and group-of-partners, are profiles developed during interactions and they are gradually built, as the agent is taking part in more and more negotiation rounds. The group profile is obtained by applying the C4.5 algorithm, and allows the classification of negotiation partners in different classes. Once these classes are

available, the agent can decide much quicker on the behavior to adopt, regarding a partner agent, than in case of the single preference profile.

The negotiation strategy is explicitly represented as a set of rules, together with the preference coefficients, associated to the rules. The preference profile is formed by these rules with coefficients, which can be fixed statically or can evolve dynamically, using a reinforcement learning algorithm. The behavior of the agents is motivated by the gain they obtain when realizing their goals or by the necessity to cooperate with other agents, in order to achieve these goals. During negotiation, the agent's beliefs on the other agents are updated, as the agent comes to know more about the others.

Because the agents' preferences are based on their interests, the preference coefficients can be modified in time. The agents can modify their preferences over negotiation outcomes, when receiving new information.

The system works in open environments, in which there is no previous knowledge about the other agents. Therefore, an agent tries to learn, little by little, during interaction, features characterizing the behavior of other agents, in a set of partners' profiles. The behavior of the agent takes into account these profiles. Also, two different approaches for updating the preference coefficients are proposed.

Chapter 6. Automated Negotiation Model using Strategies and Tactics

This chapter describes a model of heuristic negotiation between self-interested agents, which allows negotiation over multiple issues and learns the agent's negotiation strategy. The agents are using different strategies to negotiate and several models to adjust their decision during negotiation. They are capable of increasing their performance with the experience, by adapting to the environment conditions. The agents' performance, using multiple tactics, is compared to the agents having learning capabilities, based on reinforcement learning techniques. Several tests are performed, in a scenario similar to the TAC-SCM environment.

6.1. Environment Description for Automated Negotiation

The first part of this chapter presents the negotiation framework, which consists of self-interested cognitive agents, which use a set of negotiation primitives. The environment is open and the agents are able to enter and leave the environment at any time. A facilitator belongs to the system and is informed about agents' identities and abilities.

The facilitator improves agent interactions and is involved in the negotiation between agents. The facilitator can be used by the agents only to register or, additionally, to facilitate negotiation. When an agent wants to negotiate, it sends a broadcast message to all the agents in the system (first approach – as in Figure 6.1), or sends a single message to the facilitator, asking it to find other agents appropriate to negotiate (second approach – as in Figure 6.2).

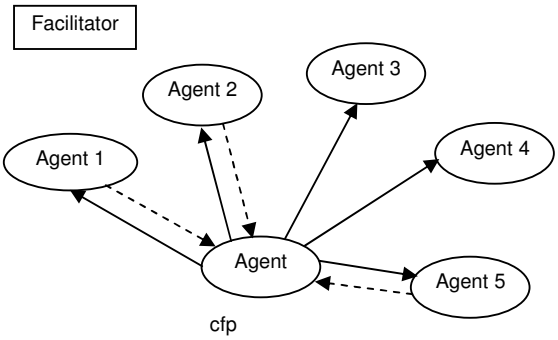


Figure 6.1. Communication between Agents – First Approach

In what follows, it is described a model of heuristic negotiation, which takes into account multi-issue negotiation, tries to adapt to the bargaining strategy of the other agent and use tactics to adapt to different situations.

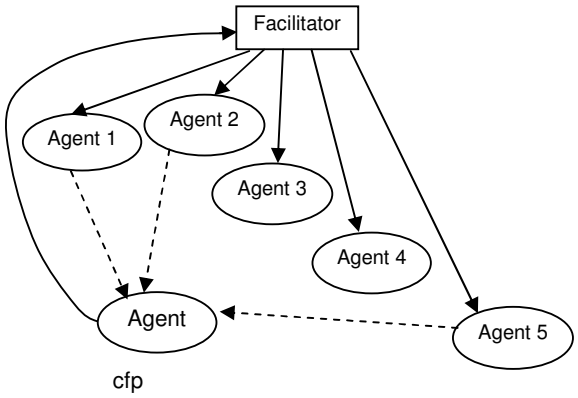


Figure 6.2. Communication between Agents – Second Approach

The approach to achieve a good behavior in a heuristic negotiation is to use utility functions, which enable an agent to generate offers and counteroffers at each step, based on different factors, such as the time, the state of a resource in the environment or the concession behavior of the opponent. The use of utility functions allows the creation of different decision strategies, which take into account the deadline of an agent, or allow the adaptation to the behavior of the negotiation partner. A tactic is applied for one issue and determines the concession behavior of an agent for this issue. A tactic is modeled as a function mapping a mental state of the agent to the domain of issues.

The negotiation framework proposed contains a multi-agent system in an open environment. The system contains buyer agents, seller agents and the facilitator, which can be used or not by the agents during negotiation. Due to the open environment, new agents can join or leave the system. An agent registers in the system when it enters, and the facilitator informs other agents regarding the new agent capabilities.

6.2. Different Approaches using Strategies and Tactics

A model of heuristic negotiation between self-interested agents, which allow bargaining over multiple issues of the negotiation object, comprise different types of negotiation primitives, including argument based ones, and a set of rules to conduct bargaining, is developed in [Florea A.M. and Kalisz E., 2007]. In order to negotiate strategically and to adapt negotiation to different partners, the agents use rewards associated to negotiation objects and the notion of regret to compare the achieved outcomes with the best possible results that could have been obtained both in a particular negotiation and in selecting the partner agent. In the current research, it is developed a set of strategies, based on weighted combination of tactics, which could be improved in time using reinforcement learning techniques.

A model of heuristic negotiation between self-interested agents is presented in [Florea A.M. and Kalisz E., 2008]. The model used negotiation objects and negotiation frames to separate the object of negotiation from the bargaining process. In order to negotiate strategically, the agents use a reinforcement learning algorithm applied on a specific state space representation of the negotiation process. The current approach has a different view upon learning, using the Q-learning algorithm in order to learn which weighted combinations of tactics used in the process of negotiation gives the highest reward and the best outcome for an agent.

A model of self-interested agents, which captures the most relevant elements of agents' behavior related to negotiation with others, is described in [Radu S. et al., 2013 a]. The agents' behavior is mainly motivated by the gain they may obtain while fulfilling their goals and negotiating. The current approach uses strategies and tactics in order to increase the agents' utilities during negotiation.

Bilateral multi-issue negotiation between self-interested agents is studied in [Fatima S.S. et al., 2006]. Three procedures are used for this process: package deal procedure, simultaneous procedure, and sequential procedure. Because of different results for each procedure, it is the system designer in charge with choosing which technique to employ in a given scenario. These procedures are used in the current approach, but the research is enhanced with different strategies.

A formal model of negotiation between autonomous agents is described in [Faratin P. et al., 1998]. This develops a set of strategies and tactics that agents could use for creating requests, evaluate proposals, and offer counterproposals. The model describes certain tactics, which agents could employ during negotiation and shows how an agent could change in time these tactics to give different forms of strategic behavior. In the current approach, combinations of tactics are used, improved in time using the Q-learning algorithm.

A negotiation strategy that describes a method to learn a model of opponent preferences in a single negotiation session is presented in [Hindriks K. et al., 2009]. The strategy should be efficient, transparent, maximizing the chance of an agreement and should avoid exploitation. In the current research, the negotiation strategy is enhanced with tactics and it is applied a learning algorithm to improve the final outcome.

A model of iterative reasoning process is developed in [Wunder M. et al., 2011], by widening the notion of a level in a hierarchy from one single strategy to a distribution over strategies, leading to a more general framework of multi-agent decision making. The current approach combines strategies and tactics, in order to learn the best possible outcome.

6.3. Negotiation Model using Negotiation Primitives

In this part, a set of negotiation primitives is presented, which are used in the negotiation model and refers to the agent x_1, initiating a bargaining with agent x_2, about a negotiation object *NO*. The proposed set of primitives are based on the ones developed in [Florea A.M. and Kalisz E., 2007], but are enhanced to deal with multi-issue negotiation objects.

A negotiation object is the range of features and issues over which agreements must be reached. The object of negotiation may be either a unique item, for example a good that the agent x_1 wants from x_2 or a service that agent x_1 offers to x_2, in case of single-issue negotiation. In case of multi-issue negotiation, the object is replaced by a negotiation package, composed of a set of negotiation objects.

The negotiation primitives are:

- $REQUEST_{x_1}$ x_2 *NO* - agent x_1 requests to agent x_2 a negotiation object (*NO*)
- $ACCEPT_{<x_1,x_2>}$ $< x_2,x_1 >$ *NO* - agent x_1 (x_2) accepts the request of x_2 (x_1) for the *NO*
- $REJECT_{<x_1,x_2>}$ $< x_2,x_1 >$ *NO* - agent x_1 (x_2) rejects the request of x_2 (x_1) for the *NO*
- $MODREQ_{<x_1,x_2>}$ $< x_2,x_1 >$ NO_1 - agent x_1 (x_2) modifies the request of x_2 (x_1) by changing some values of attributes and/or adding attributes to the *NO* to obtain NO_1
- $PACK_{x_1}$ x_2 *NO* - agent x_1 offers agent x_2 a *NO* package to be negotiated, where the *NO* package is composed by a set of objects ($NO_1, NO_2, \ldots, NO_j$)
- $SIMULTAN_{x_1}$ x_2 *NO* - agent x_1 asks agent x_2 for a multi-issue simultaneous negotiation regarding the negotiation objects *NO*, where the *NO* is composed by a set of objects ($NO_1, NO_2, \ldots, NO_j$)

With respect to each object NO_i belonging to the set of the negotiation objects *NO*, the answer of the other agent could be either *ACCEPT*, *REJECT* or *MODREQ*, but these answers are received simultaneously from the other agent.

- $SEQUENTIAL_{x_1}$ x_2 *NO* - agent x_1 asks agent x_2 for a multi-issue sequential negotiation regarding the negotiation object *NO*, where the *NO* is composed by a set of objects ($NO_1, NO_2, \ldots, NO_j$)

Regarding each object NO_i belonging to the set of the negotiation objects *NO*, the answer of the other agent could be either *ACCEPT*, *REJECT* or *MODREQ*, but these answers are received sequentially from the other agent.

The buyer agent has a strategy expressed in the form of rules. The predicates used to express this strategy are:

a) *Round* – represents the negotiation round. When a buyer or seller sends a message and then receives another, the round is increased by one;

b) *Counterproposal* – the proposal, which a buyer or seller receives from the other agent;

c) *Gain* – the gain of the buyer if he buys the product or the gain of the seller if he sells the product;

Some examples of strategy rules are presented below, where the lowercase letters denote constants and the uppercase letters represent variables:

R_1: *Round(r)* $\wedge$ *Counterproposal*(x_2, *c, NO*) $\wedge$ *Gain*(x_1, *g, NO*) $\wedge$ $c \geq g \rightarrow ACCEPT_{x_1}\ x_2\ NO$

The rule R_1 states that in the current negotiation round *r*, when the agent received a $REQUEST_{x_1}\ x_2\ NO$ message and the opponent's counterproposal is greater than or equal to the agent desired gain, then the offer is accepted.

R_2: *Step(s)* $\wedge$ *Counterproposal*(x_2, *c, NO*) $\wedge$ *Gain*(x_1, *g, NO*) $\wedge$ $c < g \rightarrow REJECT_{x_1}\ x_2\ NO$

The rule R_2 describes the case in which opponent's counterproposal is lower than the agent desired gain and the offer is rejected.

These rules can be changed, to take into account the multi-issue negotiation. For instance, denoting by all_j the situation when the negotiation is performed for all the negotiation issues included into the negotiation object, the rules R_1 and R_2 become:

R_{1m}: *Round(r)* $\wedge$ all_j *Counterproposal*(x_2, *c*, NO_j) $\wedge$ *Gain*(x_1, *g, NO*) $\wedge$ $c \geq g \rightarrow ACCEPT_{x_1}\ x_2\ NO$

R_{2m}: *Round(r)* $\wedge$ all_j *Counterproposal*(x_2, *c, NO*) $\wedge$ *Gain*(x_1, *g, NO*) $\wedge$ $c < g \rightarrow REJECT_{x_1}\ x_2\ NO$

In what follows is described an approach to design automated negotiating agents, based on a modified **alternating offers protocol**, which is improved in time using a combination of tactics and reinforcement learning techniques.

The current approach refers to multi-issue negotiation. There are three possibilities to negotiate multiple issues [Fatima S.S. et al., 2006]:

a) **Package deal** – this approach puts together all the issues and discusses them together;

b) **Simultaneous negotiation** – this involves settling the issues simultaneously, but independently of each other;

c) **Sequential negotiation** – this involves negotiating the issues sequentially, one after another.

The package deal gives the possibility to make tradeoffs between issues. These can be made when agents have different values for the issues. For instance, if there are two issues and one agent values the first more than the second, while the other values the second issue more than the first, then it is possible to make tradeoffs and to improve the utility of both agents with respect to the situation without tradeoffs. For the simultaneous and sequential approaches, the issues are settled independently and it is not necessary to have tradeoffs between agents.

The multi-issue negotiation is done with respect to the three possibilities mentioned before: package deal, simultaneous and sequential negotiation.

In the negotiation model proposed, instead of an agent that learns in the whole space of negotiation strategies, the agent is focused on a set of useful strategies [Radu S., 2013 a]. These could be learned in time, such that to choose the best strategy for negotiating with a certain agent. There are a lot of factors which can influence the result of a negotiation strategy. These factors refer to the strategies of other agents, their constraints and preferences and other characteristics of the negotiated issues.

The negotiation model contains three parts: the negotiation object, the decision making module and the negotiation protocol, as described in Figure 6.3. The negotiation object is characterized by a number of attributes for which the agents can negotiate.

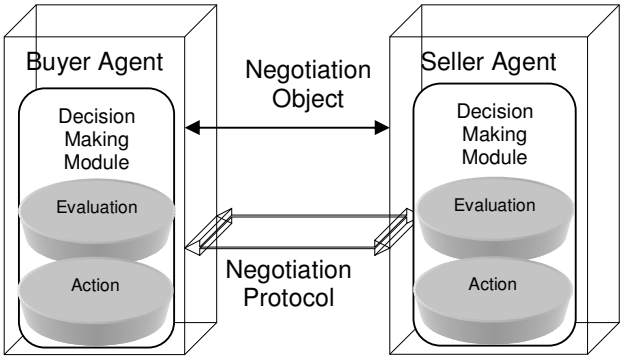


Figure 6.3. Negotiation Model Components

The decision making module consists of an evaluation part, which estimates an offer received and determines an appropriate action, and an action part, which generates and sends a counteroffer or stops the negotiation. The evaluation part is based on the fact that different values of negotiation issues have different values for negotiating agents. The value of the negotiating issues is modeled using **evaluation functions**. The higher the value of an evaluation function for a certain value of an issue is, the more suitable is that value for a negotiating agent.

An agent tries to identify the strategy, preferences and constraints of other agents. But learning a model of another agent is a complex task. Although an agent could have correct information regarding the current negotiation situation, it is necessary to find which strategy would best match the situation. It is necessary to find an approach in which agents learn what strategies to select by observing their own gains, as opposed to trying to model other agents and the state of the system.

There are two types of strategies used by the agent. The first type is similar to the alternating offers protocol and it is a built-in strategy for the agent, which doesn't use rules in it. The second one uses rules, in order to establish the best way to behave in the negotiation. These rules are stored in the internal knowledge base of the agent and are expressed in the form of first order logic predicates.

The built-in strategy is similar to the alternating offers protocol. It is supposed that the disagreement is the worst outcome, that is the agents prefer any outcome at least as much as disagreement. The agents try to maximize their utility and also the time is valuable, that is, for any outcome x and times t_1 and t_2, with $t_1<t_2$, the outcome x at time t_1 has a higher utility than outcome x at time t_2 for both agents.

The agents' offers are computed using a negotiation decision function, which determines an agent's strategy. There are **linear and non-linear strategies**. The **non-linear strategies** can be divided in: **conceder strategy**, if the agent is willing to concede a lot in the first steps of negotiation, and **boulware strategy**, if the agent is willing to concede considerably only when the time deadline is close. In Figure 6.4 is presented the boulware and conceder strategies from the point of view of the seller agent and in Figure 6.5 the same techniques are illustrated from the point of view of the buyer agent. In these figures, the price and the time are displayed in a normalized manner.

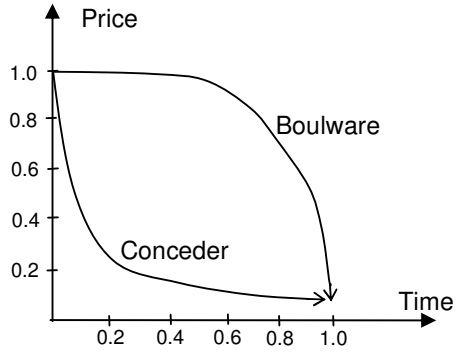


Figure 6.4. Seller Agent Strategies [Wooldridge M., 2009]

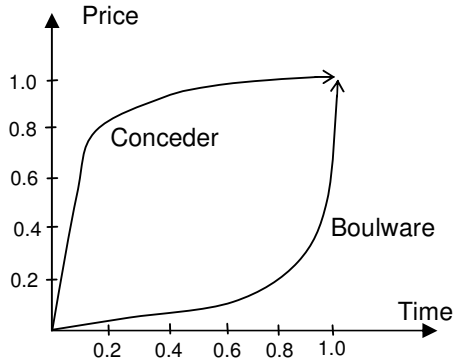


Figure 6.5. Buyer Agent Strategies [Wooldridge M., 2009]

The agent has the possibility to choose between four strategies: linear, conceder, boulware, and rule-based. For each negotiation round in which it is involved, the agent will have to compute the utility of the strategy used and will have

to fill in the values of the utility in a three dimensional matrix *M*, stored in its internal memory [Radu S. and Lungu V., 2013]. These values are used as references in future strategies. The attributes of the matrix are: the agent with whom the negotiation is performed, the utility of the strategy, the negotiation round, as presented in Figure 6.6.

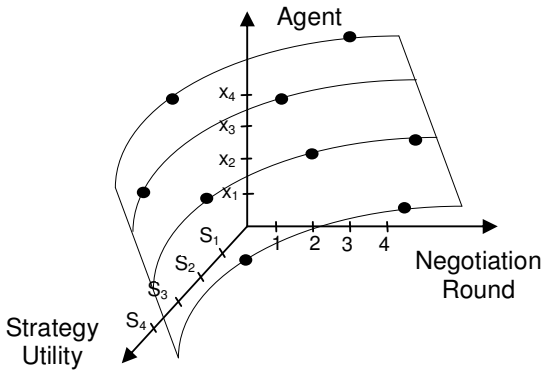


Figure 6.6. Utility Values Matrix Space

When entering the negotiation process, the short-run goal of the agent is to fill in the utilities' values in the matrix *M*. The long-run goal of the agent is to maximize its gain. For each agent x_i, with which a negotiation is performed by agent *a* in *n* rounds, using a certain strategy S_k, the utility matrix is filled with the utility values gained by the negotiation, u^a, of agent *a*. The overall utility gained after a negotiation is defined by:

$$U_a^{x_i} = \sum_{j=1}^{n} M[S_k, j, x_i] \tag{6.1}$$

If the negotiation is performed again with the same agent *a*, but now using another strategy, then the strategy which will be selected in a future negotiation will be the one that maximizes:

$$\arg\max_{S_k} \sum_{j=1}^{n} M[S_k, j, a] \tag{6.2}$$

There are two research challenges in this approach. In the first one, there is an agent, which uses four types of strategies: linear, conceder, boulware and rule-based. The agent should learn in time which strategy will give the maximum utility. For doing this, the agent builds the matrix *M* and computes the sums mentioned before, and chooses the strategy which maximizes its utility. The second challenge refers to the case in which the opponent agent knows that the other agent uses one of the four strategies previously mentioned. The goal of the opponent agent is to find which is exactly the strategy used by the other agent and how to behave in such a situation.

A possibility to identify the opponent agent strategy is to mirror each offer by making a similar one, which would implement a Tit-for-Tat like tactic. The idea of a **Tit-for-Tat strategy** in a multi-issue negotiation situation is to answer to the opponent offer with a symmetrical one. The rational negotiation strategies try to make

concession moves in a certain moment during negotiation. Using observations about the opponent agent' strategy, it is estimated and used the suitable strategy for the agent.

Suppose that *b,s* represents two negotiator agents: buyer and seller, and let *N* be the set of all possible outcomes. Each agent $a \in \{b,s\}$ has an utility function u^a which maps the outcomes to positive real numbers, that is $u^a:N\rightarrow\Re^+$. In the case of conflict, the utility is zero for both agents. In the first step, each agent has the possibility to offer, but in the next steps each agent has three options: to accept the opponent's offer, to make a new offer x^a, which is preferred by the other agent or to refuse and make a new concession and insist on its previous offer. The utility function is computed at the end of a negotiation, when a deal is obtained. Also, at each negotiation step, for each issue *j*, the buyer computes an evaluation function, which is defined on the set of current offers and takes values in the set of positive real numbers, that is $E_j^b : P \rightarrow \Re^+$. The evaluation function computes the value, which agent *b* assigns for issue *j*, between the ranges of acceptable values. The evaluation function has higher values when the agent's utility increases. The weight w_j^b represents the relative importance of the issue *j* with respect to the negotiation process. Supposing that the weights are normalized $(\sum_j w_j^b = 1)$, the agent's evaluation function for a certain proposal $x = (x_1, \ldots, x_n)$ combines the evaluation of different issues with respect to the issues' value ranges:

$$E^b(x) = \sum_j w_j^b E_j^b(x_j) \tag{6.3}$$

For each issue *j*, the evaluation function associated to that issue $E_j^b(x_j)$ is defined.

6.4. Negotiation Model using Tactics

A tactic is a function used to create a proposal value, for a certain object, based upon a given criterion. Tactics are the set of functions which determine how to compute the value of an attribute variable. Tactics can be mixed using different weights, showing the relative importance of each criterion in the strategy. The values that will be part of the proposal are computed by weighting the values proposed by each one of the tactics employed. Offers and counteroffers are generated by combinations of tactics. These generate an offer or a counteroffer for a single component of the negotiation object, using a single criterion. Different weights allow changing the importance of the criteria. For instance, when finding the values of the negotiation object, it could be more important initially to take into account the other agent's behavior than the remaining time. In this case, the tactics which enhance the behavior of other agents are preferred with respect to the ones which compute their value on the amount of the remaining time. However, because the agents are adaptive, they could change in time the importance associated to different criteria.

For instance, the remaining time could become more important than the other agent behavior, when the deadline in which an agreement should be made is close. The strategy refers to the way in which the agent changes the weights of different tactics in time.

For each object $j \in [1,\ldots,n]$ in negotiation, each agent has a range of acceptable values $[\min_j^a, \max_j^a]$, and a scoring function $V_j^a : [\min_j^a, \max_j^a] \to [0,1]$, which gives the score the agent *a* has to the value of the object *j*, in the range of its acceptable values. The higher the score, the better the agent's utility. Agents assign a weight w_j^a to each negotiation object, which represents its relative importance. Supposing that the weights are normalized, that is $\sum_{j=1}^{n} w_j^a = 1$, the agent's scoring function for a given proposal $x=(x_1,\ldots,x_n)$ combines the scores of different objects defined by the objects' value domain:

$$V^a(x) = \sum_{j=1}^{n} w_j^a V_j^a(x_j) \tag{6.5}$$

The tactics are adopted from [Faratin P. et al., 1998]:

1) **time-dependent tactics**, in which agents change their proposals when the deadline is close. If an agent has a time deadline in which an agreement should be concluded, these tactics model the fact that the agents concludes more rapidly, as the deadline approaches;

2) **resource-dependent tactics**, in which agents change their proposal with respect to the amount of available resources. These tactics model the necessity to reach an agreement, when there are limited resources. The functions for these tactics are similar to the time dependent functions, except that the domain of the function represents the quantity of resources available, instead of the remaining time;

3) **behavior-dependent tactics**, in which agents try to simulate the behavior of their opponents. The agent may choose to use imitative tactics that protect it from being exploited by other agents. In this case, the counterproposal depends on the behavior of the negotiation opponent.

Because the agents may want to consider more than one criterion to compute the value for a single issue, the generation of counterproposals is modeled as a weighted combination of different tactics defined upon the set of criteria. The values computed for different issues are the elements of the counterproposal. Suppose that an agent wants to counterpropose, taking into account two criteria: the remaining time and the previous behavior of the opponent. In this case, it can select two tactics: one from the time-dependent part and one from the behavior-dependent part. Both of these tactics give a value to counterpropose for the issue negotiated. This value is the weighted combination of the two previous values.

In time-dependent tactics, the main factor used to decide which value to offer next is the time. These tactics consist in changing the acceptance value for an issue,

depending on the remaining negotiation time. There are two different tactics in this approach. The boulware tactics maintain the offered value until the time is almost exhausted, and then it concedes up to the private value of the agent. The conceder tactics are used by agents, which go to their private value very quickly.

The resource-dependent tactics enhance the time-dependent ones. Time-dependent tactics can be seen as a kind of resource-dependent tactic, in which the only resource considered is the time. There are two types of resource-dependent tactics: dynamic-deadline and resource-estimation. Dynamic-deadline tactics represent a heuristic about how many resources are in the environment. When the resources are exhausted, the agreement must be reached. A resource to be modeled is the number of agents negotiating with a given agent and how interested they are to reach agreements. On the one hand, the greater the number of agents which are negotiating with an agent for a certain good or service, the lower the pressure on the agent to reach an agreement with any particular agent. On the other hand, as the negotiation lasts longer, the pressure on the agent to reach an agreement increases. The resource-estimation tactics generate counterproposals depending on how a particular resource is being consumed. The agent should become more conciliatory, as the quantity of resource decreases. When the quantity of the resource is almost zero, the agent concedes up to its private value for the issue negotiated. When there are a lot of resources, a more boulware behavior is expected.

The behavior-dependent tactics compute the next offer based on the previous attitude of the negotiation opponent. The main difference between the tactics in this category is in the type of imitation they perform. One category imitates proportionally, another in absolute terms, and the last one computes the average of the proportions in a number of previous offers. In relative Tit-for-Tat, the agent replicates, in percentage terms, the behavior that its opponent performed some steps ago. In random absolute Tit-for-Tat, the behavior is the same as in relative Tit-for-Tat, but in absolute terms. In the averaged Tit-for-Tat, the agent computes the average of percentages of changes of its opponents' history, when determining its next offer.

A tradeoff tactic finds a proposal with the same utility as the previous one offered, but expecting to be more acceptable for the opponent agent. For the buyer agent *b*, which receives a proposal *y* from the seller agent *s*, the tactic allows agent *b* to choose a new proposal *x'* to offer to agent *s*, which fulfills two conditions: the new proposal *x'* has the same utility as the offer previously proposed *x*, and the new proposal *x'* is the most similar to the offer *y* proposed by *b*.

The tactics give a possibility to adapt to different situations, considering certain resources. The initial proposal has an important role in negotiation. Choosing the initial value could be learned in time. Using different weighted combinations of tactics, in order to find the suitable one in each negotiation, gives the agent an adaptive behavior for obtaining increased gains.

The current approach uses a reinforcement learning algorithm to combine tactics and to learn good outcomes. In particular, it is used a version of the Q-

learning algorithm [Srivihok A. and Sukonmance P., 2005]. The **Q-learning algorithm** estimates the value of executing an action in each state. Actions are vectors of weighted combinations of tactics, used in the process of negotiation. The Q-learning algorithm converges to the optimal combinations of state-action pairs, after each action has been tried enough times. The ranking is due to rewards by matching actions to certain states. The agent performs the next steps for each issue:

1. *determines the current state for each issue;*
2. *chooses a weighted combination of tactics;*
3. *uses this weighted combination of tactics for the next proposal;*
4. *computes the rewards obtained.*

When the Q-values associated with each state-action pair are updated, this means rewarding the actions which give good results. The Q-learning formula used for updating is:

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma * \max_a Q(s',a') - Q(s,a)] \quad (6.6)$$

where α is the learning rate, r is the reward obtained by executing action a in state s, γ is the discount factor, $max_a Q(s',a')$ is the maximum Q value for the actions in the next state.

The Q-learning algorithm is applied on tactics. In this case, the action a represents a vector of weights of tactics, which could be applied in a given state s. The Q-learning chooses, for each issue, the highest scored weighted combination of tactics. Because the environment is dynamic, the same action may not lead to a desired output, when applied in the same state.

The actions which result in deals are rewarded with a function depending on the deal values utility and on the average utility. This allows determining the deals corresponding to higher utilities. The failure of a goal has a negative reward on the action performed.

The action selected to be performed is the action with the highest Q-value. In dynamic environments, actions do not give always the same results. In order to obtain a high reward, the agent should prefer actions that were considered good in the past, but in order to find them, it must try actions that were never selected before.

This leads to the tradeoff between exploitation and exploration. In order to fulfill this tradeoff, two policies are possible: the ε-greedy approach, which selects uniformly, with a probability ε, a non-greedy action; the softmax policy, which uses a given degree of exploration T, for choosing between non-greedy actions, while considering their ranking.

It is better that the agent does not prefer the first action leading to a deal, such that the agent increases the utility obtained in deals. Before the agent tries enough actions, it has an incomplete knowledge of the environment. The agent should know what action to perform, for obtaining a deal, but not what the best actions are. In order to put the agent to try all the actions available before preferring the best ones, it is used a reinforcement learning technique called optimistic initial values. It means that

all the Q-values associated with the actions are initialized to a value greater than the expected reward. This measure increases the initial action exploration, because the Q-values are updated to lower values.

In this approach, it is used the ε-greedy exploration, which selects a random action with probability ε and the best action, which is the one with the highest Q-value, with probability 1-ε. In this way, it can be seen as defining a probability vector over the action set of the agent for each state. Suppose that $z=(z_1, z_2, \ldots, z_j)$ is one of these vectors. The probability z_i of doing action i is given by:

$$z_i = \begin{cases} (1-\varepsilon)+(\varepsilon/n), & \text{if Q of } i \text{ is the highest} \\ \varepsilon/n, & \text{otherwise} \end{cases} \quad (6.7)$$

where n is the number of actions in the set.

6.5. Simulation Scenarios Discussion

The multi-agent system is tested in a B2B framework, which is defined in a similar manner to the TAC-SCM (Trading Agent Competition – Supply Chain Management) environment [Groves et al., 2009]. The settings consist of agents that buy components from other agents, assemble these components and sell the assembled results to customers. All the three types of agents are artificial in the environment. The agents encountered uncertainty and incomplete information. Different initial values are used, to obtain a variety of strategic behaviors. Also, the agent is adjusted with the initial parameters, which produce good performance, according to the measurements.

The agents assembling the components may use reinforcement learning and different strategy rules, or, alternately, the tactics types. The assembling agents are behaving in such a way as to obtain a maximum gain. The goal of the agents is to forecast the demand, which will give the highest overall profit. Depending on the behavior of other agents, the agent focuses on exactly enough demand, in order to achieve its goals. To this aim, the agents use behavior-dependent tactics, combined with time-dependent tactics, which are further combined with resource-dependent tactics.

A prototype of the system is implemented in Jade, in an environment in which there are n assembling agents, m supplier agents and p customer agents. The values for n and p are varied between 2 and 5 agents, and m between 2 and 10 agents. The best behavior and output is obtained if the agents use negotiation rules and a combination of tactics.

Many computation parameters kept in the state have unknown factors and many approximations are done. The available capacity of each seller is approximated, because it helps to determine the profitability of a given part, produced by a certain seller. The approximation is done by estimating three other parameters, which are relevant to that seller: the delay of the seller when the previous component

is delivered, the prices offered by that seller for previous components, the estimation of partial offers, given by that seller until now. Then, the amount of available computers in the inventory is approximated, because it is not profitable to produce computers out of components in the inventory, if there are no pending orders. However, the amount of computers that can be delivered to buyers is estimated, according to the components in the inventory. In order to do that, an algorithm that converts a collection of components to computers is used. The algorithm is greedy, trying to construct as many computers as possible, starting with the most valuable ones. Each computer received a grade, according to the relevant information. For instance, if there is an increased demand for a certain computer, than that computer receives a better grade. When determining the grades, the computer type is also taken into account.

Requesting more components doesn't guarantee that it is possible to obtain them. Other agents could have higher priority or some sellers couldn't handle the demand. Trying to obtain more customers' orders is challenging, when competing against good opponents, which can offer better prices. The length of the simulation changes the suitable tactics. The agent should have a plan for the long run, constructing its inventory to last for the whole simulation.

The chosen strategy coordinates decisions. An agent's profit can be influenced by the market competition. For instance, if other agents are focusing on a large part of the market, then an agent should decrease its prices in order to reach its target.

The relative percentage of each product, which is the product mixture, used to fulfill the goal, is easily changed after each negotiation round, based upon the profit of each product type. The exact percentage of each product in the mixture is equal to the percentage of the agent's total profit it had on the previous negotiation round. When the profit of a product increases or decreases, with respect to the profit of other products, then its percentage in the product mixture increases or respectively decreases.

In the case when a certain product has no profit, the product mixture is computed in a similar manner. But the strategy is changed, such that the quantity of the product which has no profit is decreased. This makes the demand to decrease. Eventually, the agent changes its heuristic in order to finish the simulation with small stocks in the inventory.

The agent learns the combination of tactic weights for a negotiation round. During transitions between states in which the agent is bargaining, the reward is set to zero for each issue's weighted combination of tactics. This doesn't affect the conceder or boulware behavior. For a negative reward, the agent has fewer proposals to obtain a deal. When reaching a state representing a deal, the reward is equal to the sum of the agent's utility for each issue. The deal configuration influences the reward. If a deal is not reached, it means the weighted combination of tactics is not good enough.

Also, the multi-agent system is tested in a multi-issue negotiation, which simulates the negotiation between a telecommunication service provider and

customers. The telecommunication company offers services like: Internet connection, short and long distance phone calls and TV cable channels. In this case, there are three issues which could be negotiated: Internet services, phone calls services and TV services. There are different packages, which include different types of services:

a) the Internet service has different characteristics, like: bandwidth available, wireless capabilities, firewall protection;

b) the phone service offers: short distance calls, long distance calls, and international calls;

c) the TV cable service offers different channels, which include national channels, international channels, film channels, music channels, entertainment channels.

In the negotiation with a telecommunication company, in the case of the package deal negotiation, if the Internet service has a weight of 0.5, the phone service has a weight of 0.2 and the TV cable service has a weight of 0.3, then the evaluation function is computed as:

$$E^b(x) = \sum_{j=1}^{3} w_j^b E_j^b(x_j) = 0.5 * E_1^b(x_1) + 0.2 * E_2^b(x_2) + 0.3 * E_3^b(x_3) \quad (6.8)$$

6.6. Conclusion

This chapter presented how an agent can learn to improve its negotiation strategy. The idea is that the agent has a set of useful negotiation strategies, from which to choose the best one. The space of all possible negotiation strategies is very big. Restricting the agent's behavior to a relevant set of negotiation strategies reduces the space the agent needs to learn.

The chapter showed that it is possible to design agents with different negotiation strategies that have good performances in dynamic environments. The knowledge obtained in past negotiations can be an important advantage in certain scenarios.

The package deal is the optimal procedure for each agent. However, the package deal could have a higher complexity than the other two procedures from the computational point of view and gives Pareto optimal results, as opposed to the simultaneous and the sequential procedures.

The negotiation primitives that are proposed may be extended to take into account arguments. Each argument type defines preconditions for its usage. If these are fulfilled, then the agent can use the argument. The arguments types that are foreseen are similar to those presented in [Kraus S. et al., 1998]. The selection of arguments should be also coupled with the utility function of the agent.

A future research direction is to use anticipatory genetic algorithms for learning tactics behaviour. This approach will extend the work on anticipatory genetic algorithms done in [Mocanu I. et al., 2010].

Chapter 7. A Negotiation Model with BDI Agents

This chapter develops a conceptual model of self-interested agents in a multi-agent system, which takes into account several facets of agent knowledge and behavior: abilities, history of relations with other agents, cooperation and negotiation options. This framework is based on the **BDI** (Beliefs-Desires-Intentions) model and uses rules for encoding the negotiation strategy. The inference rules that guide negotiation are based on price. This conceptual model considers several profiles, providing information about the object and the type of the current cooperation request and on the cooperation history. The gradually refinement of the cooperation profiles is seen as a form of agent learning. The model allows the definition of several types of agents, by changing their behavior, according to the desire to develop good cooperation relations with other agents in the system or the desire to obtain the maximum gain. This model also introduces the notions of utility for negotiation objects and for the roles the facilitator has in the negotiation process.

7.1. An Overview of the JADE Platform

JADE (**J**ava **A**gent **De**velopment Framework) is a software environment implemented in **Java** language [Tudose C. et al., 2013], aiming at the development of multi-agent systems that comply with **FIPA** (**F**oundation for **I**ntelligent **P**hysical **A**gents) specifications [Bellifemine F. et al., 2007]. JADE provides many of the basic classes required for agent based software development. Some of them are:

- *Agent*;
- *Behaviour*;
- *ACLMessage*;
- *Ontology*.

JADE simplifies the agents' development, while ensuring standard compliance through a comprehensive set of system services and agents. JADE provides the following components for agent's management [Bellifemine F. et al., 2007]:

- **AMS** (Agent Management System), which besides providing white page services, as specified by FIPA, it also plays the role of authority in the platform;
- **DF** (Directory Facilitator) provides yellow pages services to other agents;
- **ACC** (Agent Communication Channel), which provides a Message Transport System (**MTS**) and is responsible for sending and receiving messages on an agent platform.

Eclipse is the **IDE** (Integrated Development Environment) commonly used to develop the JADE application. It is easy to integrate Eclipse with JADE, so that when the agent application is executed, it runs JADE and deploys the agent into the runtime environment.

In JADE, a **behavior** represents a task that an agent can carry out. Each such behavior class must implement two abstract methods. The *action()* method defines the operations to be performed, when the behavior is in execution. The *done()* method returns a boolean value, to indicate whether or not a behavior has completed and must be removed from the pool of behaviors an agent is executing. To make an agent execute the tasks represented by a behavior object, the behavior must be added to the agent by means of the *addBehaviour()* method of the *Agent* class.

The JADE platform provides a yellow pages service which allows any agent to dynamically discover other agents at a given point in time. A specialized agent called the **DF** (Directory Facilitator) provides the yellow pages service in JADE. Using this service any agent can both register (publish) services and search for (discover) services [Bellifemine F. et al., 2007].

Agent communication is a fundamental feature of JADE and is implemented with respect to the FIPA specifications. The JADE communication paradigm is based on **asynchronous message passing**. Each agent is equipped with an incoming message box and message polling can be blocking or non-blocking. A message in JADE is implemented as an object of the *jade.lang.acl.ACLMessage* class, which is then calling the *send()* method of the *Agent* class.

One of the most useful tools to use when developing a multi-agent system with JADE is the **Sniffer Agent**. This is another agent built into JADE, which allows the user to see the message interactions taking place in real time. The arrows show the type of message, the sender, the receiver and when it was sent within the lifetime of the system. If more information is required about any of the ACL messages, the user can double click the specific arrows and full details are displayed.

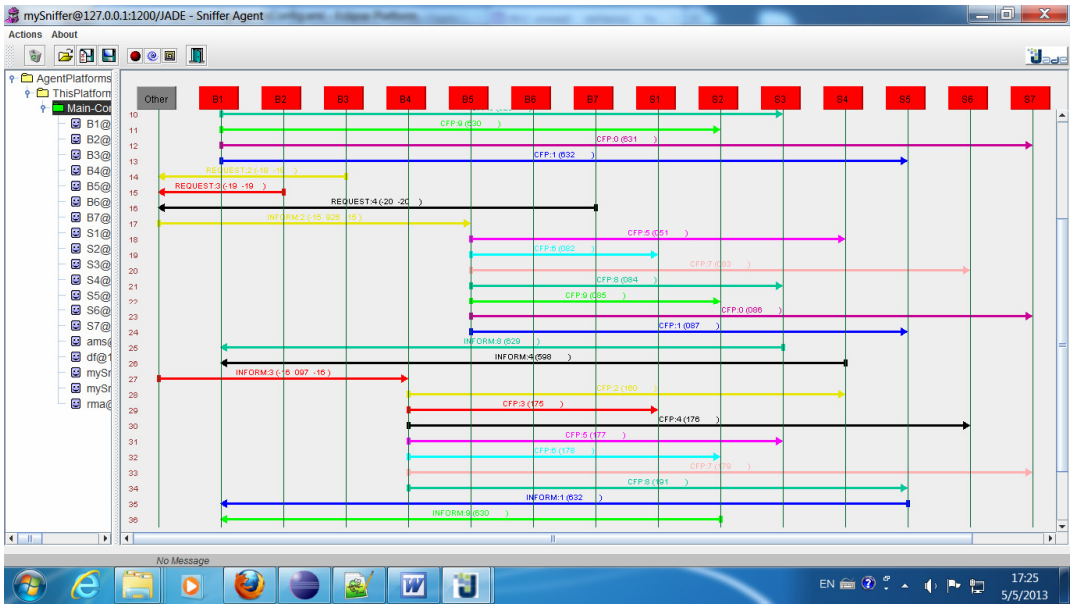


Figure 7.1. Screen Capture Showing the Multi-Agent System in Action

Figure 7.1 represents a screen capture of the JADE **Sniffer Agent** for the multi-agent system implemented, showing the messages exchange between agents during a many-to-many negotiation.

7.2. Agent Model

A model of self-interested agents, acting in a multi-agent system, is described in this chapter. The system is inspired by the BDI approach. More details about the BDI model are given in the next paragraphs.

Beliefs represent knowledge on the environment and are a way of representing the state of the world. Beliefs are important, because the environment is dynamic and the system has only a local view of the world. As beliefs represent possibly imperfect information about the environment, the semantics of the belief component should be in harmony with belief logics, even though the computational representation is not symbolic or logical.

Desires or goals are another component of the system. A goal represents some desired final state. In computational terms, a goal may be the value of a variable, a structure, or a symbolic expression in some logic.

Intentions represent committed plans or procedures. In computational terms, intentions may be a set of executing threads in a process, which can be interrupted when receiving feedback from the possibly changing environment.

The main parts of a system designed for a dynamic and uncertain environment should contain representations of beliefs, desires, intentions, and plans, that is a BDI agent [Georgeff M. et al., 1999]. A BDI agent has a set of beliefs (***B***), desires (***D***), and intentions (***I***). A BDI agent has a set of percepts ***p***, by means of which it recognizes an event, which can be either a state change or an action occurrence. The percept obtained from the environment may cause a change in beliefs, leading to a belief revision process, as presented in Figure 7.2.

The reasoning components of a BDI agent are denoted by functions:

a) $brf: B \times p \rightarrow B$ is the belief revision function, where *p* represents the set of percepts;
b) $options: B \times D \times I \rightarrow I$ is the function which weights competing alternatives to achieve the desires and decides the course of action to be taken;
c) $plan: B \times I \rightarrow I\Pi$ is the function that structures intentions into plans.

The developed model takes into account different profiles, providing information about the object and the type of the current negotiation request and of the previous negotiations. This model presents the notions of utility for negotiation objects and for the roles that the facilitator could have in negotiation. It is supposed that the agents have consistent desires, so these may be considered equal to their goals.

When a goal is fulfilled, the agent develops one or more plans for achieving it. Planning refers at constructing sequences of actions. The agent itself could perform part of these actions, while others are not part of its abilities. So, the agent has to

negotiate the actions with other counterparts in the system. The agent could also decide to delegate to others, parts of the actions it is able to perform. So, the agent includes the actions in a plan in two distinct classes: **"intentions-to"** (actions it is able and willing to perform) and **"intentions-that"** (actions the agent does not know how or does not want to perform). The later is the negotiation object with the others in the system.

An agent has a set of inference rules for realizing the set of goals, for updating the mental state and for plan generation. After the plan creation, the agent analyses the intentions for realizing the plan and identifies the intentions-to and intentions-that, by investigating its abilities. In order to fulfill the intentions-that, the agent has to negotiate their achievement with the other agents in the system. For having an efficient negotiation, the agent is endowed with a set of negotiation inference rules and also for the evaluation of the cooperation profile of other agents, another set of inference rules.

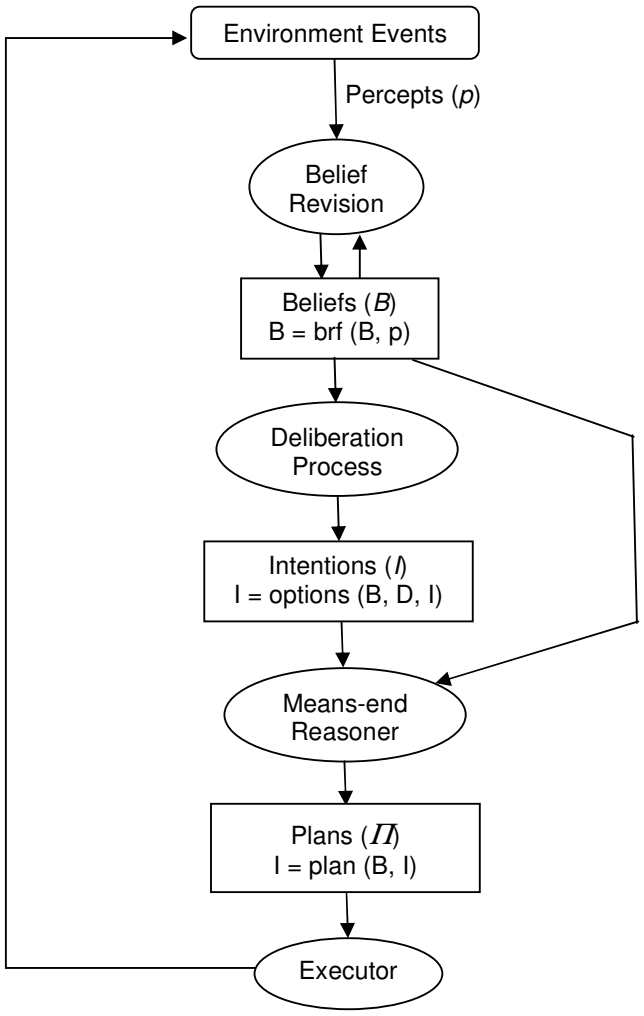


Figure 7.2. The BDI Agent Model [Shoham Y. and Leyton-Brown K., 2009]

Negotiation is based on the gain the agent obtains from fulfilling its goals, but negotiation criteria contain also the **cooperation profile** the agent has developed to describe previous interactions with other agents in the system. The cooperation profile, which is always updated, may be considered as part of the agent's **beliefs** about the others in the system.

Each agent has different reasoning capabilities, described by inference rules. These are updating the beliefs about the agents, are constructing efficient plans for goal fulfillment, and are conducting successful negotiation. An agent should have details on others' identity and abilities. It does not need to keep information on all the agents, but only on those it is interacting with, successfully or not, along its activity. In the system there is the **facilitator**, which supports interactions between agents.

All agents start their activity by a registration message that informs the facilitator on their names and on the actions they can perform or want to receive. An agent can address the facilitator for different purposes: extension of its beliefs about agents able to perform a given action, selection of the best candidate to perform the requested action or negotiation with other agents. The facilitator has a **selection role** and also a **negotiation role**. During negotiation, the facilitator applies a protocol of type **Contract Net**, as opposed to agent negotiation, in which the negotiation protocol and strategy may be defined in different ways.

The control structure of the agent has two steps. The first step refers to the control of independent agent's activities, while the second step concerns negotiation and reaching agreements. The first step contains goal selection, plan generation, and analysis of intentions in the generated plan. Each goal has an associated gain. The gain associated to intentions is used by the agent during negotiation.

The agent analyzes if the actions are in the current range of its abilities. This analysis may conduct to a revision of plan generation or of goal selection. The agent identifies the intentions-to and the intentions-that. The intentions-that are analyzed to identify the agents which, according to the agent beliefs, are able to perform them. If there are intentions-that which cannot be satisfied by other agents, the agent will revise its plans or its goals.

The first step of the agent's control structure is shown in Table 7.1.

The second step refers to negotiation, when the agent has identified intentions-that necessary to fulfill its goals. Now, the agent tries to reach an agreement with other agents to perform these actions. Negotiation may be between two or more agents. In case of multi-agent negotiation, the inference rules indicate the agent with which other agents negotiate.

The second step of the agent's control structure is in Table 7.2.

The negotiation inference rules generate and select the suitable request and the agents to which the requests are sent. The answers to these requests are evaluated and, in case a counterproposal is received, the new conditions may

either be accepted, rejected with a justification, or subject to a new counterproposal.

Table 7.1. Step I for Agent A

Select **goals** {*Goals*} as a subset of {$Desires_A$}
Generate **plans** for selected goals {$Plans_{Goal_A}$}
Analyse **actions** in {$Plans_{Goals_A}$} with respect to agent's **abilities** {$Abilities_A$}
If there are **intentions-that then** identify the agents {*i*} with {$Abilities_i$} able to do **intentions-that**
If no such agents exist **then revise** {$Plans_{Goals_A}$} or {$Goals_A$}
Perform all **intentions-to**

Table 7.2. Step II for Agent A

Generate and send requests for agents in {*i*} to do **intentions-that**
Evaluate answers, accept them or generate counterproposals
Evaluate incoming requests {$Request_A$} and generate answers
Update the mental model and the **cooperation profile**
Send answers to {$Request_A$}

After each negotiation step, the mental model is updated. This refers to beliefs, intentions, goals, and to the cooperation profile of the agents with which it has been exchanging messages.

An agent, as presented in the Figure 7.3, has an input communication component, which analyzes the incoming communication primitives. These contain usually a proposal, an acceptance or rejection of a previous proposal. A proposal is stored in the knowledge base of the agent for future reference. Proposals or rejections go into a primitive evaluation and generation component, which makes a decision about whether to accept, reject or generate a counterproposal, or even terminate the negotiation. Then, the output communication primitives component sends the response to the other agent.

Also, an agent maintains a knowledge base of its mental attitudes, such as beliefs, desires, intentions [Wooldridge M., 2002], as well as models of the environment and the negotiation counterparts. This knowledge may be used in the evaluation and generation of proposals by judging the validity and worth of proposals made, for instance, by verifying whether proposals are actually feasible and do not conflict with the current observations of the environment. Moreover, the knowledge base is updated when new information arrives.

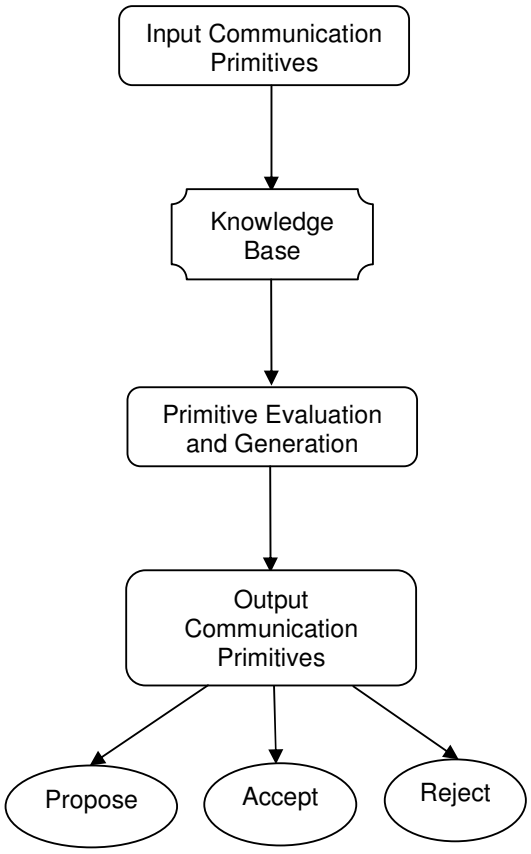


Figure 7.3. The Elements of a Negotiating Agent

Cognitive aspects associated to negotiation activities should be taken into account. One feature is the **reflection capacity**, because in the end of each bargaining, it would be important for an agent to evaluate how efficient was the interaction with other agents, identifying the positive points and the negative ones, by computing the gain associated to the transaction. Another feature is the **empathy**, which means the attempt to understand the real necessities or preferences of other agents, in order to decrease the divergences between them. This is fulfilled by changes in the cooperation profile, during the negotiation with a certain agent.

Also, the agent has the capacity to do many negotiations at the same time, because it encounters different agents, which have different proposals.

The system has an open and flexible architecture, where agents can be viewed as sellers and buyers in parallel negotiations. A negotiator is an autonomous entity. The agent registers in the negotiation environment and also informs about the services supplied, in case it is a seller.

The **JESS** engine represents the agent's inference engine, which stores the knowledge base of the business domain and also contains the negotiation strategies for the agent, the facts and the concepts of bargaining.

JADE is used as an infrastructure for building the multi-agent system and JESS as a mechanism to provide the inference engine for the agents that are negotiating.

The negotiation strategies show how the agent reasons at each moment of the bargaining. Each agent possesses different negotiation strategies, encoded in rules.

When an agent receives proposals, it filters the best proposal from its point of view and chooses it. The use of rules and the learning techniques for the negotiation strategies make the agent more flexible to changes that can occur in its strategies or objectives. This can happen for several reasons, like changes in the business domain, the necessity to consider new information related to the business or the necessity to act on new business domains.

The reasoning model of the agent is presented in Figure 7.4. The agent first generates its desires, based on its beliefs and individual internal motivations. Then, it generates possible plans for achieving its desires. It selects the best plan, based on some appropriate criteria, which becomes an intention. If the agent can execute its intention on its own, then it would do so. Otherwise, if the agent needs to negotiate with another agent in order to contract out parts of its intention, it would initiate a negotiation dialogue with a certain agent. If the negotiation results in a deal, then the agent can execute its intention. The agent ends the negotiation if it decides that no deal is reachable. Until then, during the negotiation process, the agent may update its beliefs and planning knowledge, as a result of receiving new information, which updates its desires and intentions.

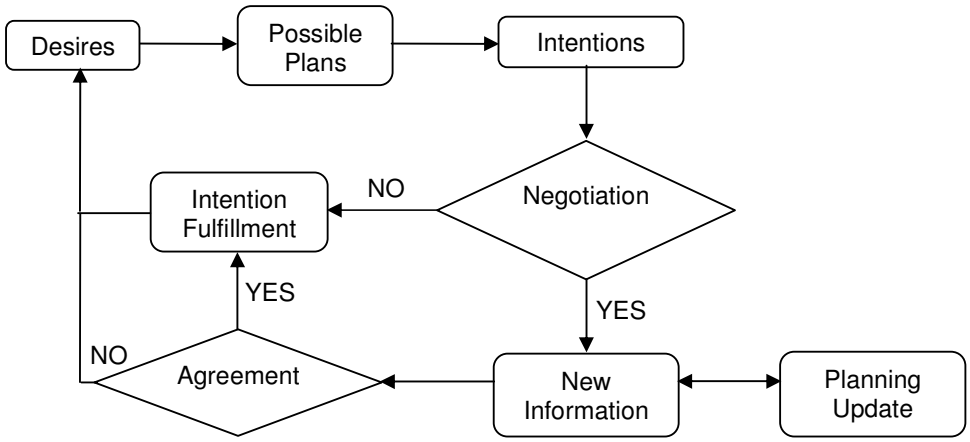


Figure 7.4. The Reasoning Model of the Agent

7.2.1. Automated Negotiation Design

The agents that appear in the negotiation process are the buyers and the sellers. The negotiation mechanism is based on the following ideas:

a) The buyers and the sellers are represented by software agents;

b) The negotiation strategy of the agents is expressed using rules;

c) The knowledge base consists of a set of facts and a number of rules;

d) Both buyers and sellers use the same negotiation protocol;

e) When an agent receives an offer from another agent, it stores the new facts in the knowledge base. Consequently, the control part activates the inference engine, which in turn updates the knowledge base with the inference result, according to the strategy rules. Finally, the control part retrieves the result and presents it to the communication part of the agent.

The architecture of a negotiating agent is presented in the Figure 7.5.

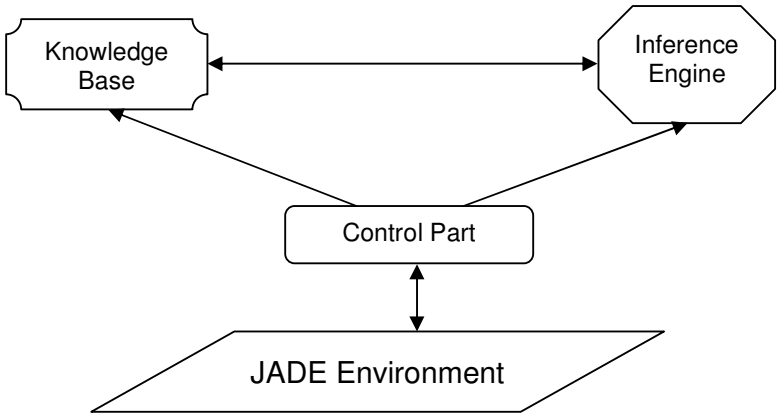


Figure 7.5. The Negotiating Agent Architecture

The three types of agents that are identifiable in the system are: the buyer, the seller, and the facilitator. This has a secondary role and the agents use it to find each other and to register what goods or services they want to buy or sell. The requirements of the buyer are represented using rules and priorities. These include both mandatory requirements that must be fulfilled, and also preferences, which can be used to select among the offers possible to be accepted. These requirements are communicated to the facilitator agent by the buyer agent. When the facilitator receives a request, it matches this to the offers, by running the request specification against the available offers. Then, the requester's preferences are applied to select the most suitable one, which is then presented to the agent making the request.

The multi-agent system architecture is presented in Figure 7.6.

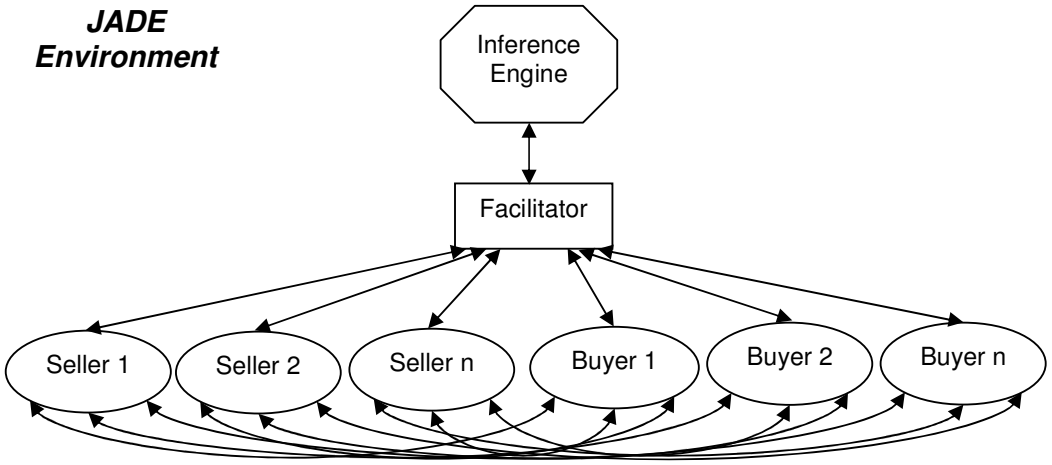


Figure 7.6. Multi-Agent System Architecture

7.2.2. Agent Control Structure

The control structure of the agent is composed of two phases. The first phase is dedicated to the control of agent's activities which doesn't depend on other agents, while the second phase is dedicated to negotiation and reaching agreement. The steps for the first phase are:

1. *Generate desires, based on beliefs;*
2. *Generate candidate plans for achieving desires;*
3. *Generate intention, as the best possible plan;*
4. ***If** it has capabilities **then** execute intention*;
5. ***If** it hasn't capabilities **then** negotiate.*

The steps for the second phase are:

1. *Receive offers from other agents;*
2. *Update beliefs and planning knowledge;*
3. *Update desires and intentions;*
4. *Generate counteroffers and send them to the other agents.*

In the implemented system, there is a **base agent class**, extended by buyer and seller agents, which contains common functions used by all agents. The main features of this class are described in the following steps.

1. *Read negotiation object;*
2. *Check if the current agent has products with the desired attributes;*
3. ***If** there are no products in stock **then** **REJECT-PROPOSAL**;*
4. *Check message type;*

5. ***If*** *message=***CALL-FOR-PROPOSAL** (only sellers receive CFP) ***then*** *get all products of that type;*
    5.1. ***If*** *there are less products than the other agent wants to buy* ***then*** **REJECT PROPOSAL***;*
    5.2. ***If*** *there are offers* ***then*** *send them;*
6. ***If*** *message=***INFORM** (only buyers receive INFORM) ***then***
    6.1. ***If*** *multiple products received* ***then*** *find which one is the best;*
    else *check* ***if*** *the attributes' values match the request;*
7. ***If*** *message=***PROPOSE** ***then*** *process offer;*
8. ***If*** *message=***AGREE** ***then***
    8.1. Add information to statistics file and compute gain;
    8.2. Remove product from stock;
    8.3. Collect statistics when negotiation ends;
9. ***If*** *message=* **REJECT-PROPOSAL** ***then*** *end negotiation with reject.*

Each **buyer agent** does the following steps:

1. *When the buyer enters negotiation, send to all sellers* **REQUEST***'s for the products of interest;*
2. *Every 5 seconds the buyer agent sends to all sellers* **REQUEST***'s for the products of interest;*
3. *Get the agents which sell what the agent needs to buy.*

Each **seller agent** performs the following steps, when processing an **ACCEPT** offer:

1. *Remove the product from the stock;*
2. ***If*** *OK* ***then*** *send* **AGREE** *message to the buyer agent;*
3. *Add the negotiation results to statistics and compute the gain;*
4. *Remove the current bid from the list of open bids*;
5. *Reject the offers of other agents interested in this product;*
6. *Add to statistics file the result of the failed negotiations.*

7.2.3. Negotiation Protocol

The message exchange protocol employed by the agents is described in the following algorithm:

1. *When seller agents are initialized, they inform the* ***DF*** *which products they sell. The products are registered by an alias used both by buyers and sellers;*
2. *Each 5 seconds the buyer agent (****B****) uses* ***DF*** *to find which agents are selling the first product on its list of products to buy (****P_1****);*
    2.1. ***If*** *there are no agents selling product* ***P_1*** ***then B*** *tries to find the agents which sell product* ***P_2*** *and so on;*

2.2. ***If B*** *finds an agent (or more) which sells the product* ***P_1 then*** *sends* **CFP***;*

2.3. ***If*** *no agents are found* ***then B*** *repeats the procedure after 5 seconds;*

3. *Seller agent (**S**) receives* **CFP** *message sent by **B**;*
4. ***S*** *finds out all products having the requested alias;*
5. ***S*** *composes an* **INFORM** *message and add to it all products found;*
6. ***S*** *sends the* **INFORM** *message to **B**;*
7. ***B*** *receives the* **INFORM** *message sent by **S**;*
8. ***B*** *validates the product attributes* ***if*** *the validation rule file is defined for the product;*
9. *After the products are validated* ***then B*** *chooses the best product;*
10. ***B*** *sets the quantity of products it wants to buy;*
11. ***B*** *computes the classification for **S** in a cooperation class;*
12. ***B*** *uses the rules defined for the product business model to get the new price;*
13. ***B*** *sends* **PROPOSE** *message to **S**;*
14. ***S*** *receives* **PROPOSE** *message from **B**;*
15. ***S*** *uses the rules defined for its product business model to evaluate the new offer;*
16. ***If S*** **ACCEPT***s the offer* ***then***

    16.1. ***S*** *updates the product list;*

    16.2. ***If*** *stock > negotiated quantity* ***then*** *stock is decreased;*
    else *product is removed;*

    16.3. ***If*** *there are enough products in stock to fulfill the request* ***then S*** *sends* **AGREE** *message to **B**;*

    16.4. *Add information to statistics file;*

    16.5. *Remove this bid from the list of open bids;*

    16.6. *Reject the offers of other agents interested in this product;*

    16.7. *Add to statistics file the result of the failed negotiations;*

    16.8. *End current negotiation;*

    16.9. ***If*** *there are no more products like this* ***then S*** *informs* ***DF*** *to deregister this product from its services;*

17. ***If S*** **REJECT***s the offer* ***then*** *negotiation ends;*

    17.1. *Statistics file is updated;*

18. ***If S*** *has a new price* ***then S*** *sends* **PROPOSE** *message to **B**;*
19. ***B*** *receives* **PROPOSE** *message from **S**;*
20. ***B*** *uses the rules defined for its product business model to evaluate the new offer.*

The multi-agent system collects the following information in the statistics file:

a) **Negotiation Statistics** – for each agent, the system collects the total gain, the number of negotiations, and the total number of negotiation rounds;

b) **Cooperation Statistics** – for each agent and partner cooperation class, the system collects the number of negotiation rounds, the number of negotiations, the total gain, and the list of agents included in that cooperation class;
c) **Supply Demand Statistics** – supply is computed as the sum of quantities of all seller products and demand is computed as the sum of quantities of all buyer products. Their ratio is computed by dividing the demand to the supply.

7.3. Negotiation Objects Utility

The **negotiation object** (NO) is described by a unique **name** and a number of **attributes**. Because negotiation is a dynamic process, implying changes of the initial conditions for the negotiation object, the NO structure is flexible. It allows not only changes of attribute values, but also the addition of new ones. A negotiation object has two classes of attributes: **dynamic** attributes, which can be negotiated, and **fixed** attributes, which cannot be modified during negotiation. Each attribute has a **name**, a **value**, which can be of different types, and a **flag** showing if the attribute may be modified or not during the negotiation process. Figure 7.7 displays the attributes associated to the negotiation object.

Figure 7.7. The Attributes Associated to the Negotiation Object

A **negotiation object utility** estimates how useful a NO is for the negotiator. It allows the agent to compare the initial NO, which has the utility +1, with modified negotiation objects, either by itself during negotiation, or by the other agent in counterproposals. During one or several negotiation steps, some attribute values of the initial NO are changed, and some attributes are added, as a result of appropriate requests. The utility of modified negotiation objects is less than the initial utility, because it is supposed that the negotiator has increased the value of NO, by adding an attribute to it. The formula to compute a negotiation object utility, $\mathbf{U_O(NO)}$, is:

$$U_O(NO) = \sum_{i=1}^{n} w_i(V_i^0 - V_i)/V_i^0 + \sum_{j=1}^{m} w_j e(V_j) \quad (7.1)$$

where:
n - number of modified attributes;
m - number of added attributes;
V_i^0 - value of the *i*-th attribute;

w_i - weight of attribute *i* in the utility of NO;
e - function to capture how the utility increases or decreases by adding an extra attribute; for decreasing the utility $w_j < 0$.

For an attribute that cannot be modified during negotiation, for instance, a time limit that cannot be extended due to plan constraints, w_i will have a big negative value, so that the utility of the negotiation object becomes very low. Utility values under a given threshold leads to unacceptable objects and to a new step in negotiation or to an unsuccessful negotiation.

7.4. Modeling the Facilitator

An agent asking the facilitator should give information about the role it wants the facilitator to assume and about the requested action. This information must be expressed for both a selection and a negotiation role of the facilitator, and is contained in the **request profile**. The request profile has two components:

a) the **negotiation object profile**, dealing with the action of interest;

b) the **facilitator role profile**, referring to the role played by the facilitator in the multi-agent system.

The negotiation object profile should be designed in relation to the facilitator role profile, providing the appropriate information, which is represented by the name of the NO, accompanied or not by a subset or by all of its attributes.

The facilitator role profile refers to the role the facilitator is requested to have, but also to the content of the expected answer, that is the list of agents able to perform the requested action.

The roles developed by the facilitator can be divided in:

Role 1 - Inform Role - the facilitator is requested to inform the agent issuing the request which agents are capable to fulfill the action specified in the negotiation object profile. In this case, upon receiving the facilitator's answer, the source agent selects the agent/agents with which it would negotiate and start negotiation to reach agreement;

Role 2 – Selection Role - the facilitator sends the request received from the source agent to other agents and returns the list of agents interested in starting negotiation, which will be conducted by the source agent itself;

Role 3 – Negotiation Role - the facilitator has the selection role, assuming afterwards the negotiation, under the conditions reflected in the negotiation object profile. The negotiation could be of two different types: **single-party**, with the best candidate found, or **multi-party**. In the second case, the facilitator sends copies of the negotiation object profile to all agents able to perform the corresponding action. There is an answer reception deadline, as measured by a timer set when sending the copies. From the answers received in time, the facilitator selects the best one, and sends the result back to the source agent.

The model used for selecting facilitator roles is described in Figure 7.8.

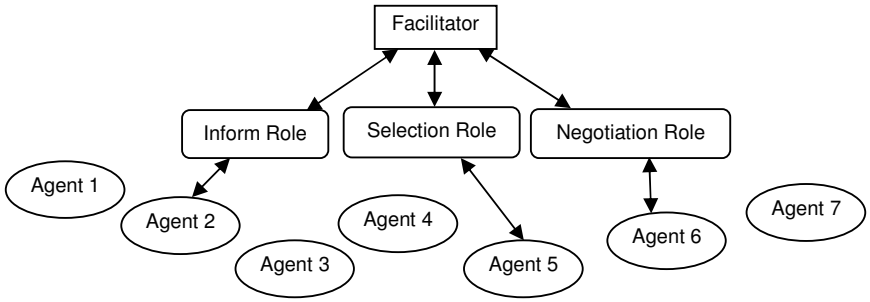


Figure 7.8. The Facilitator Roles

7.5. Negotiation Primitives

The negotiation process implies an exchange of information between agents **A** and **B**. The process is started by the negotiator **A**, which issues a request for a product, this being directed towards agent **B**. Agent **B** may accept the request, may reject it, or may modify the request by changing the value of an attribute of the NO, or by adding a new attribute. In the later two cases, agent **A** has to decide upon accepting, rejecting or modifying the counterproposal, based on its intentions, goals and built plans. Negotiation may continue by performing several consecutive steps, during which one or the other agent modifies the NO, until a deal has been concluded or the negotiation failed.

The **negotiation primitives** used in this scenario are:

a) **(Request NO)** - request of a negotiation object - the entire negotiation object structure is transmitted;

b) **(Accept name(NO))** - accept the request for the NO;

c) **(Reject name(NO))** - reject the request for the NO;

d) **(ModReq name(NO) assign(NO,X,V_1))** - counterproposal to modify the current NO by assigning a different value V_1 to attribute X;

e) **(ModReqAdd name(NO) assign(NO,Y,V) Yes/No)** - counterproposal for modifying the NO by adding an extra attribute Y, with value V, subject to further modification (**Yes**) or not (**No**).

The negotiation primitives to address the facilitator and to ask it for a particular role are the following:

a) **(Inform NO_profile one/all)** - the agent asks the facilitator to inform it about one agent (**one**) or all the agents (**all**) capable of performing NO or interested in NO (facilitator's **Role 1**); in this case NO_profile = name(NO);

b) **(Select NO_profile one/all)** - the agent asks the facilitator to find one or all the agents willing to start a negotiation on object NO (facilitator's **Role 2**); the NO_profile may contain only the name of the NO or the entire structure of the NO or a subset of that structure;

c) **(Negotiate NO_profile one/all)** - the agent asks the facilitator to bargain the

negotiation object NO for it (facilitator's **Role 3**); the NO_profile contains the entire NO structure and, optionally, acceptable ranges for the negotiation object attributes; the facilitator will return the partner agent with which it concluded a successful negotiation or all the agents interested in NO.

When performing a negotiation role, the facilitator has no information about the goals and plans of the agents on behalf of which it negotiates. So, it can't use a heuristic mechanism to decide on the suitability of a received modified request or to issue other modified negotiation objects, as is the case of a negotiating agent. Instead, it uses the **Contract Net Protocol** to choose among bidding agents the one with the best offer, or the best offer within acceptable ranges, in case the NO profile specifies acceptable ranges for the NO's attributes. If the negotiation object is a service offered by the agent, this indicates it is interested in all agents wishing this service with the "**all**" parameter in the **Negotiate** primitive.

To implement the **Contract Net Protocol**, the facilitator uses the following negotiation primitives, which are understood by the other agents in the system:

a) **(Call NO)** – the facilitator calls for proposals on NO and the entire NO is transmitted;

b) **(Bid NO)** – the agents in the system are bidding for NO;

c) **(Allocate name(NO))** – the facilitator allocates the NO to a bidding agent and informs it accordingly;

d) **(Refuse name(NO))** – the facilitator informs a bidding agent that its bid is rejected.

The negotiation primitives described above are included in a send or receive message:

a) (**Send** Sender Receiver negotiation_primitive)

b) (**Receive** Receiver Sender negotiation_primitive)

Figure 7.9 displays the messages exchange using the **Contract Net Protocol**, between the initiator of the communication and the participant.

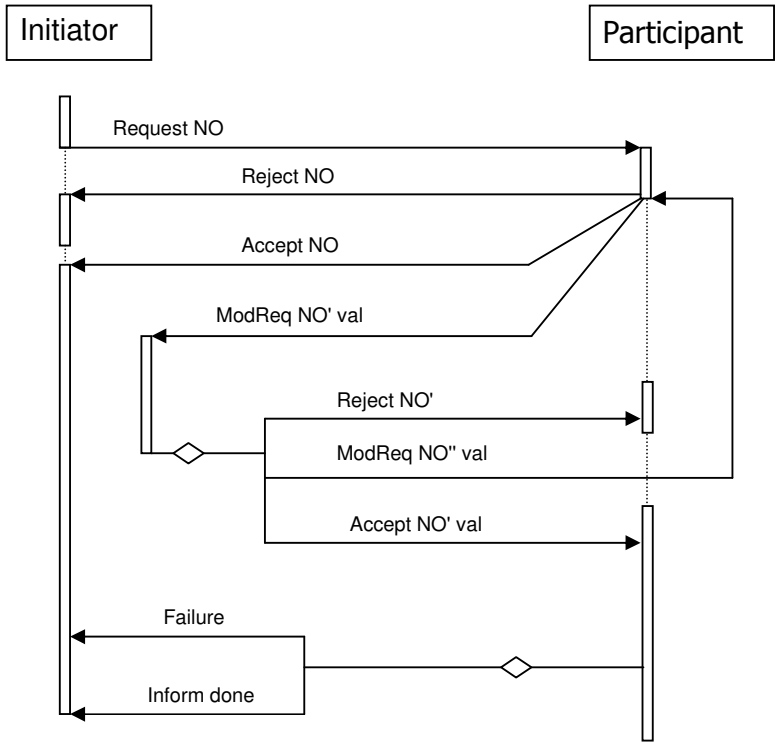


Figure 7.9. Messages Exchange using the Contract Net Protocol

7.6. Multi-Agent Cooperation

A negotiation object has a utility, computed at the end of a successful negotiation. The utility of a negotiation is defined using the following formula:

$$U(Neg_{NO}) = \frac{U(NO)}{w * Ns} \tag{7.2}$$

where *w* represents the weight of the negotiation length, 0< w ≤1, and *Ns* the number of negotiation steps.

Each agent increases its knowledge on the system by taking into account the results obtained when trying to cooperate with other agents.

The **individual cooperation profile (A,X)** is developed by the agent **A** for each agent **X** it was involved in a negotiation, considering also the role played by the facilitator in each negotiation process. The individual cooperation profile is defined by the following set of data:

a) agent name **X**;

b) **X**'s abilities;

c) cooperation synthesis for each facilitator role, stored in four fields:

1) s_i - the total number of successful negotiations with *X* under role r_i of the facilitator;

2) $GU_s(r_i)$ - global negotiation objects utility, intended to measure how much the

agent has gained by successful negotiations when the facilitator assumed role r_i;

$$GU_s(r_i) = \frac{\sum_{j=1}^{s_i} U^{r_i}(NO'_j)}{s_i}, i = 1,3 \quad (7.3)$$

where NO'_j is the modified object concluded by the deal established in negotiation *j*;

3) f_i - the total number of unsuccessful negotiations with *X* under role r_i of the facilitator;

4) $GU_f(r_i)$ - global negotiation objects utility, intended to measure how much the agent has lost, because of unsuccessful negotiations with *X*, when the facilitator assumed role r_i.

$$GU_f(r_i) = \frac{\sum_{j=1}^{f_i} U^{r_i}(NO_j)}{f_i}, i = 1,3 \quad (7.4)$$

where NO_j is the object negotiated in the unsuccessful negotiation *j*.

The Figure 7.10 displays the **individual cooperation profile (A,X)** for agent **A**.

Figure 7.10. The Individual Cooperation Profile for Agent A

The choice of the facilitator's role in the negotiation process is based on additional information, concerning **cooperation profile of a group of agents**. Such profiles include the same fields as the individual cooperation profile described above, with the only difference that the first field is either **all** or a list of agent names.

After each negotiation, an agent updates both the individual and the group cooperation profiles, as well as the cooperation profiles of the groups having as member the negotiation partner. When there are few interactions with another agent, the agent uses, most of the time, the group profile. As the number of interactions with an agent grows, it is guided by the individual profile or by a combination of the existing profiles.

The facilitator's role is chosen according to the maximum gain principle. It requires to compute the gain obtained for each role r_i, i=1,3:

$$Gain(r_i) = GU_s(r_i) - GU_f(r_i) \quad (7.5)$$

and choose the facilitator role that gives the maximum gain:

$$\max_k (Gain(r_k)) \quad (7.6)$$

In the beginning, the agent does not have too much information in the

cooperation profiles and therefore it chooses one or the other roles of the facilitator arbitrarily. As interactions go on, information is accumulated in these profiles, quicker in the group cooperation profile and slower in the individual cooperation profile. The emergence of these profiles is seen as a form of agent learning, in which the agent learns how to deal with the group of agents and then gradually refines its beliefs towards dealing with individual agents.

The tradeoff between exploitation and exploration should be considered. If after successful deals under role r_i of the facilitator, the agent starts prefering only this role, then it may ignore some opportunities. Therefore, it may be added an exploratory coefficient ε, $\varepsilon \geq 1$, and choose role r_k such that:

$$\max_k (\varepsilon * Gain(r_k)) \tag{7.7}$$

7.7. Conclusion

This chapter presented a conceptual model of cognitive agents in a multi-agent system, based on the BDI model, which considers goals equal to desires. While cooperating with other agents, an agent develops cooperation profiles, which are gradually refined, and uses them to control its behavior during negotiation. A particular agent, the facilitator, has a special role in the system. It is responsible with the management of agents and their related abilities.

The contribution of this chapter is to provide a conceptual model for agent cooperation, based on several profiles. The request profile, with its two components, the negotiation object profile and the facilitator role profile, provides information about the object and the type of the current cooperation request addressed to the facilitator, while the cooperation profiles provide synthetic information on the cooperation history. The gradually refinement of the cooperation profiles is seen as a form of agent learning.

This model also introduces the notions of utility for negotiation objects and for the roles the facilitator could play in the negotiation process. The later allow the use of a very simple selection mechanism for the facilitator's role.

The developed model can be extended to take into account the emotions of the agents. In this case, the affective behavior of the agents can be designed using an emotional modeling architecture [Lungu V. et al., 2013]. This emphasizes emotional reasoning in the context of other forms of reasoning, specific to the BDI model.

Also, the norms in the multi-agent system influence the agents' reasoning. Norms have an important role in the agents' society [Trascau M. et al., 2013]. They tend to be generally accepted by all the agents in the system, as the norms are the result of a complex process of emergence, which starts with the simplest interactions between agents and continues as the agents negotiate repeatedly in the framework of a certain scenario.

Chapter 8. Automated Negotiation for the Travel Agency Business Model

8.1. Travel Agency Automated Negotiation Rules

This chapter presents a business scenario involving a travel agency [Radu S., 2013 b]. A person, represented by the buyer agent, wants to book the hotel for the holiday. The user has several criteria, each attribute having a certain priority for the user, upon which he/she decides the hotel to choose. The buyer agent representing the human should know the criteria and the user's preferences. The criteria are presented in Table 8.1 and the priorities for the user are displayed, on a 1 to 10 scale. These characteristics are encoded in the **XML configuration file** associated to this business model. The content of the configuration file is read into the application using the **SAX** (Simple API for XML) **parser**. SAX provides a mechanism for reading data from an XML file.

Table 8.1. Attributes of the Travel Agency Negotiation Scenario

Requirements for the Hotel	Priority
Location in the Town	10
Close to the Railway Station	8
Close to the Airport	5
Close to the Bus Station	7
Close to the Touristic Area	10
Hotel Classification	9
View from the Room	6
Room Type (single, double, twin, appartment)	8
Private/Shared Bathroom	10
Balcony	7
TV	8
Wireless Internet Access	5
Refrigerator	8
Type of Meals	10
Indoor/Outdoor Swimming Pool	8
Gym Room	4
Air Conditioning in the Room	10

The rules are defined in **Jess**. There are several rules, upon which the negotiation is performed. There are different rules defined for each type of communication primitive. The higher priority is associated to the **ACCEPT** rules, the medium priority to the **REJECT** rules, and the lower priority to the **PROPOSE** rules.

It follows a description of the negotiation rules for the travel agency negotiation scenario, first for the buyer agent, and then for the seller agent. This scenario

accomodates one-to-one, one-to-many, many-to-one, and many-to-many negotiations.

For the buyer, regarding the **ACCEPT** communication primitive, there are the following rules:

1) The buyer will accept an offer *o* from a seller, which is either non-cooperative or unknown, for a certain quantity of *q* items, each having the price between *minPrice* and *maxPrice*, which is less than *q*(minPrice+maxPrice)/2*;

```
(defrule accept1
  (declare (salience 100))
  (NegotiationObject {sellerClassification == "nc" ||
sellerClassification == "u"}
                     (currentOffer ?o) (quantity ?q))
  (Product (minPrice ?minPrice) (maxPrice ?maxPrice))
  (test (<= ?o (* ?q (/ (+ ?minPrice ?maxPrice) 2))) => (add (new Offer
"accept1" ?*accept*)))
```

2) The buyer will accept an offer *o* from a seller, which is either slightly cooperative or cooperative, for a certain quantity of *q* items, each having the price between *minPrice* and *maxPrice*, which is less than *q*(maxPrice-(minPrice+maxPrice)/2)*;

```
(defrule accept2
  (declare (salience 100))
  (NegotiationObject {sellerClassification == "sc" ||
sellerClassification == "c"}
                     (currentOffer ?o) (quantity ?q))
  (Product (minPrice ?minPrice) (maxPrice ?maxPrice))
  (test (<= ?o (* ?q (- ?maxPrice (/ (+ ?minPrice ?maxPrice) 2)))))
  => (add (new Offer "accept2" ?*accept*)))
```

3) The buyer will accept an offer *o* from a seller, which is either very cooperative or highly cooperative, for a certain quantity of *q* items, each having the price between *minPrice* and *maxPrice*, which is less than *q*maxPrice*;

```
(defrule accept3
  (declare (salience 100))
  (NegotiationObject {sellerClassification == "vc" ||
sellerClassification == "hc"}
                     (currentOffer ?o) (quantity ?q))
  (Product (minPrice ?minPrice) (maxPrice ?maxPrice))
  (test (<= ?o (* ?q ?maxPrice)))
  => (add (new Offer "accept3" ?*accept*)))
```

4) The buyer will accept an offer *o* from a seller, which is either non-cooperative or unknown, for a certain quantity of *q* items, each having the price between *minPrice* and *maxPrice*, which is less than *q*(minPrice+10)*;

```
(defrule accept4
  (declare (salience 100))
  (NegotiationObject {sellerClassification == "nc" ||
sellerClassification == "u"}
                     (currentOffer ?o) (quantity ?q))
  (Product (minPrice ?minPrice))
```

```
      (test (<= ?o (* ?q (+ 10 ?minPrice))))
      => (add (new Offer "accept4" ?*accept*)))
```

5) The buyer will accept an offer *o* from a seller, which is either slightly cooperative or cooperative, for a certain quantity of *q* items, each having the price between *minPrice* and *maxPrice*, which is less than *q*(minPrice+15)*;

```
     (defrule accept5
      (declare (salience 100))
      (NegotiationObject      {sellerClassification      ==      "sc"      ||
sellerClassification == "c"}
                          (currentOffer ?o) (quantity ?q))
      (Product (minPrice ?minPrice))
      (test (<= ?o (* ?q (+ 15 ?minPrice))))
      => (add (new Offer "accept5" ?*accept*)))
```

6) The buyer will accept an offer *o* from a seller, which is either very cooperative or highly cooperative, for a certain quantity of *q* items, each having the price between *minPrice* and *maxPrice*, which is less than *q*(minPrice+20)*;

```
     (defrule accept6
      (declare (salience 100))
      (NegotiationObject      {sellerClassification      ==      "vc"      ||
sellerClassification == "hc"}
                          (currentOffer ?o) (quantity ?q))
      (Product (minPrice ?minPrice))
      (test (<= ?o (* ?q (+ 20 ?minPrice))))
      => (add (new Offer "accept6" ?*accept*)))
```

For the **REJECT** communication primitive of the buyer, there are the following rules:

1) The buyer will reject an offer for a negotiation object, received after a certain time threshold, which is defined as *5* seconds for each negotiation. The message *Negotiation time elapsed* is sent from the buyer to the seller;

```
     (defrule reject1
      (declare (salience 50))
      (NegotiationObject (timeElapsed ?te))
      (test (>= ?te ?*maxNegTime*))
      => (add (new Offer "reject1" ?*reject* "Negotiation time elapsed.")))
```

2) The buyer will reject the current offer *o* for a certain amount of *q* items, received from a non-cooperative or unknown seller, having the price greater than *maxPrice+20*, when the negotiation step is greater than *5*. The message *Price too high* is sent from the buyer to the seller;

```
     (defrule reject2
      (declare (salience 50))
      (NegotiationObject      {sellerClassification      ==      "nc"      ||
sellerClassification == "u"}  {step > 5}
                          (currentOffer ?o) (quantity ?q))
      (Product (maxPrice ?maxPrice))
      (test (>= ?o (* ?q (+ ?maxPrice 20))))
      => (add (new Offer "reject2" ?*reject* "Price too high")))
```

3) The buyer will reject the current offer *o* for a certain amount of *q* items, received from a slightly cooperative or cooperative seller, having the price greater

than *maxPrice+40*, when the negotiation step is greater than *5*. The message *Price too high* is sent from the buyer to the seller;

```
(defrule reject3
 (declare (salience 50))
 (NegotiationObject {sellerClassification == "sc" ||
sellerClassification == "c"} {step > 5}
                    (currentOffer ?o) (quantity ?q))
 (Product (maxPrice ?maxPrice))
 (test (>= ?o (* ?q (+ ?maxPrice 40))))
 => (add (new Offer "reject3" ?*reject* "Price too high")))
```

4) The buyer will reject the current offer *o* for a certain amount of *q* items, received from a very cooperative or highly cooperative seller, having the price greater than *maxPrice+60*, when the negotiation step is greater than *5*. The message *Price too high* is sent from the buyer to the seller;

```
(defrule reject4
 (declare (salience 50))
 (NegotiationObject {sellerClassification == "vc" ||
sellerClassification == "hc"} {step > 5}
                    (currentOffer ?o) (quantity ?q))
 (Product (maxPrice ?maxPrice))
 (test (>= ?o (* ?q (+ ?maxPrice 60))))
 => (add (new Offer "reject4" ?*reject* "Price too high")))
```

For the **COUNTERPROPOSE** communication primitive of the buyer, there are the following rules:

1) The price offered by a buyer, in the first negotiation step, for a certain quantity *q* of products, having the price between *minPrice* and *maxPrice*, to a non-cooperative or unknown seller, is equal to *q*(minPrice-5)*;

```
 (defrule firstPrice1
 (declare (salience 10))
 (NegotiationObject {step == 1} {sellerClassification == "nc" ||
sellerClassification == "u"} (quantity ?q))
 (Product (minPrice ?minPrice))
 => (add (new Offer "firstPrice1" (* ?q (- ?minPrice 5)) ?*propose*)))
```

2) The price offered by a buyer, in the first negotiation step, for a certain quantity *q* of products, having the price between *minPrice* and *maxPrice*, to a slightly cooperative or cooperative seller, is equal to *q*minPrice*;

```
(defrule firstPrice2
 (declare (salience 10))
 (NegotiationObject {step == 1} {sellerClassification == "sc" ||
sellerClassification == "c"} (quantity ?q))
 (Product (minPrice ?minPrice))
 => (add (new Offer "firstPrice2" (* ?q ?minPrice) ?*propose*)))
```

3) The price offered by a buyer, in the first negotiation step, for a certain quantity *q* of products, having the price between *minPrice* and *maxPrice*, to a very cooperative or highly cooperative seller, is equal to *q*(minPrice+10)*;

```
(defrule firstPrice3
 (declare (salience 10))
```

```
      (NegotiationObject {step == 1} {sellerClassification == "vc" ||
sellerClassification == "hc"} (quantity ?q))
      (Product (minPrice ?minPrice))
      => (add (new Offer "firstPrice3" (* ?q (+ 10 ?minPrice))
?*propose*)))
```

The buyer agent has three different strategies used during negotiation. It can employ either the linear strategy, the conceder strategy or the boulware strategy. There are different rules used for each strategy.

For the **linear strategy** of the buyer, there are the following rules:

1) The buyer will increase its previous offer *o* with *2*, to the non-cooperative or unknown sellers, when the negotiation step is greater than *1*;

```
     (defrule newPrice4
      (declare (salience 10))
      (NegotiationObject {step > 1} (previousOffer ?o)
{sellerClassification == "nc" || sellerClassification == "u"}) (Strategy
{strategy == ?*linear*})
      =>(add (new Offer "newPrice4" (+ ?o 2) ?*propose*)))
```

2) The buyer will increase its previous offer *o* with *4*, to the slightly cooperative or cooperative sellers, when the negotiation step is greater than *1*;

```
     (defrule newPrice5
      (declare (salience 10))
      (NegotiationObject {step > 1} (previousOffer ?o)
{sellerClassification == "sc" || sellerClassification == "c"}) (Strategy
{strategy == ?*linear*})
      =>(add (new Offer "newPrice5" (+ ?o 4) ?*propose*)))
```

3) The buyer will increase its previous offer *o* with *6*, to the very cooperative or highly cooperative sellers, when the negotiation step is greater than *1*;

```
     (defrule newPrice6
      (declare (salience 10))
      (NegotiationObject {step > 1} (previousOffer ?o)
{sellerClassification == "vc" || sellerClassification == "hc"}) (Strategy
{strategy == ?*linear*})
    => (add (new Offer "newPrice6" (+ ?o 6) ?*propose*)))
```

For the **conceder strategy** of the buyer, there are the following rules:

1) The buyer will increase its previous offer *o* with *25*, to the non-cooperative or unknown sellers, when the negotiation step is greater than *1* and lower than *6*;

```
      (defrule newPrice7
      (declare (salience 10))
      (NegotiationObject {step > 1 && step < 6} (previousOffer ?o)
{sellerClassification == "nc" || sellerClassification == "u"})
      (Strategy {strategy == ?*conceder*})
      => (add (new Offer "newPrice7" (+ ?o 25) ?*propose*)))
```

2) The buyer will increase its previous offer *o* with *35*, to the slightly cooperative or cooperative sellers, when the negotiation step is greater than *1* and lower than *6*;

```
     (defrule newPrice8
      (declare (salience 10))
      (NegotiationObject {step > 1 && step < 6} (previousOffer ?o)
{sellerClassification == "sc" || sellerClassification == "c"})
```

```
      (Strategy {strategy == ?*conceder*})
      => (add (new Offer "newPrice7" (+ ?o 35) ?*propose*)))
```

3) The buyer will increase its previous offer *o* with *45*, to the very cooperative or highly cooperative sellers, when the negotiation step is greater than *1* and lower than *6*;

```
     (defrule newPrice9
      (declare (salience 10))
      (NegotiationObject {step > 1 && step < 6} (previousOffer ?o)
{sellerClassification == "vc" || sellerClassification == "hc"})
      (Strategy {strategy == ?*conceder*})
      => (add (new Offer "newPrice7" (+ ?o 45) ?*propose*)))
```

4) The buyer will increase its previous offer *o* with *5*, to the non-cooperative or unknown sellers, when the negotiation step is greater than *6*;

```
     (defrule newPrice10
      (declare (salience 10))
      (NegotiationObject {step >= 6} (previousOffer ?o)
{sellerClassification == "nc" || sellerClassification == "u"}) (Strategy
{strategy == ?*conceder*})
      => (add (new Offer "newPrice10" (+ ?o 5) ?*propose*)))
```

5) The buyer will increase its previous offer *o* with *10*, to the slightly cooperative or cooperative sellers, when the negotiation step is greater than *6*;

```
     (defrule newPrice11
      (declare (salience 10))
      (NegotiationObject {step >= 6} (previousOffer ?o)
{sellerClassification == "sc" || sellerClassification == "c"}) (Strategy
{strategy == ?*conceder*})
      => (add (new Offer "newPrice11" (+ ?o 10) ?*propose*)))
```

6) The buyer will increase its previous offer *o* with *15*, to the very cooperative or highly cooperative sellers, when the negotiation step is greater than *6*;

```
     (defrule newPrice12
      (declare (salience 10))
      (NegotiationObject {step >= 6} (previousOffer ?o)
{sellerClassification == "vc" || sellerClassification == "hc"}) (Strategy
{strategy == ?*conceder*})
    => (add (new Offer "newPrice12" (+ ?o 15) ?*propose*)))
```

For the **boulware strategy** of the buyer, there are the following rules:

1) The buyer will increase its previous offer *o* with *0.1*, when the negotiation step is greater than *1* and the time elapsed in the negotiation is less than *4* seconds, represented by the global variable *boulwareTime1*;

```
      (defrule newPrice13
      (declare (salience 10))
      (NegotiationObject {step > 1} (previousOffer ?o) {timeElapsed <
?*boulwareTime1*})
      (Strategy {strategy == ?*boulware*})
      => (add (new Offer "newPrice13" (+ ?o 0.1) ?*propose*)))
```

2) The buyer will increase its previous offer *o* with *10*, when the negotiation step is greater than *1* and the time elapsed in the negotiation is greater than *4* seconds, represented by the global variable *boulwareTime1*;

```
(defrule newPrice14
 (declare (salience 10))
 (NegotiationObject {step > 1} (previousOffer ?o) {timeElapsed >=
?*boulwareTime1*})
 (Strategy {strategy == ?*boulware*})
=> (add (new Offer "newPrice14" (+ ?o 10) ?*propose*)))
```

3) The buyer will increase its previous offer *o* with *15*, when the negotiation step is greater than *1* and the time elapsed in the negotiation is greater than *4.5* seconds, represented by the global variable *boulwareTime2*;

```
(defrule newPrice15
 (declare (salience 10))
 (NegotiationObject {step > 1} (previousOffer ?o) {timeElapsed >=
?*boulwareTime2*})
 (Strategy {strategy == ?*boulware*})
 => (add (new Offer "newPrice14" (+ ?o 20) ?*propose*)))
```

For the seller, regarding the **ACCEPT** communication primitive, there are the following rules:

1) The seller will accept an offer from a non-cooperative or unknown buyer, for a certain quantity of *q* products, having the price between *minPrice* and *maxPrice*, with the price greater than *q*(minPrice+maxPrice)/2;*

```
 (defrule accept1
 (declare (salience 100))
 (NegotiationObject {buyerClassification == "nc" ||
buyerClassification == "u"} {step > 0}
                    (currentOffer ?o) (quantity ?q))
 (Product (minPrice ?minPrice) (maxPrice ?maxPrice))
 (test (>= ?o (* ?q (/ (+ ?minPrice ?maxPrice) 2))))
 => (add (new Offer "accept1" ?*accept*)))
```

2) The seller will accept an offer from a slightly cooperative or cooperative buyer, for a certain quantity of *q* products, having the price between *minPrice* and *maxPrice*, with the price lower than *q*(maxPrice-(minPrice+maxPrice)/2);*

```
(defrule accept2
 (declare (salience 100))
 (NegotiationObject {buyerClassification == "sc" ||
buyerClassification == "c"} {step > 0}
                    (currentOffer ?o) (quantity ?q))
 (Product (minPrice ?minPrice) (maxPrice ?maxPrice))
 (test (<= ?o  (* ?q (- ?maxPrice (/ (+ ?minPrice ?maxPrice) 2))))
  => (add (new Offer "accept2" ?*accept*)))
```

3) The seller will accept an offer from a very cooperative or highly cooperative buyer, for a certain quantity of *q* products, having the price between *minPrice* and *maxPrice*, with the price lower than *q*maxPrice;*

```
(defrule accept3
 (declare (salience 100))
 (NegotiationObject {buyerClassification == "vc" ||
buyerClassification == "hc"} {step > 0}
                    (currentOffer ?o) (quantity ?q))
 (Product (minPrice ?minPrice) (maxPrice ?maxPrice))
```

```
  (test (<= ?o (* ?q ?maxPrice)))
  => (add (new Offer "accept3" ?*accept*)))
```

4) The seller will accept an offer from a non-cooperative or unknown buyer, for a certain quantity of *q* products, having the price between *minPrice* and *maxPrice*, with the price greater than *q*(minPrice+30);*

```
(defrule accept4
  (declare (salience 100))
  (NegotiationObject      {buyerClassification      ==      "nc"      ||
buyerClassification == "u"} {step > 0}
                      (currentOffer ?o) (quantity ?q))
  (Product (minPrice ?minPrice))
  (test (>= ?o (* ?q (+ 30 ?minPrice))))
  => (add (new Offer "accept4" ?*accept*)))
```

5) The seller will accept an offer from a slightly cooperative or cooperative buyer, for a certain quantity of *q* products, having the price between *minPrice* and *maxPrice*, with the price greater than *q*(minPrice+20);*

```
(defrule accept5
  (declare (salience 100))
  (NegotiationObject      {buyerClassification      ==      "sc"      ||
buyerClassification == "c"} {step > 0}
                      (currentOffer ?o) (quantity ?q))
  (Product (minPrice ?minPrice))
  (test (>= ?o (* ?q (+ 20 ?minPrice))))
  => (add (new Offer "accept5" ?*accept*)))
```

6) The seller will accept an offer from a very cooperative or highly cooperative buyer, for a certain quantity of *q* products, having the price between *minPrice* and *maxPrice*, with the price greater than *q*(minPrice+10);*

```
(defrule accept6
  (declare (salience 100))
  (NegotiationObject      {buyerClassification      ==      "vc"      ||
buyerClassification == "hc"} {step > 0}
                      (currentOffer ?o) (quantity ?q))
  (Product (minPrice ?minPrice))
  (test (>= ?o (* ?q (+ 10 ?minPrice))))
  => (add (new Offer "accept6" ?*accept*)))
```

For the **REJECT** communication primitive of the seller, there are the following rules:

1) The seller will reject an offer for a negotiation object, received after a certain time threshold, which is defined as *5* seconds for each negotiation. The message *Negotiation time elapsed* is sent from the seller to the buyer;

```
(defrule reject1
  (declare (salience 50))
  (NegotiationObject (timeElapsed ?te))
  (test (>= ?te ?*maxNegTime*))
  => (add (new Offer "reject1" ?*reject* "Negotiation time elapsed.")))
```

2) The seller will reject the current offer *o* for a certain amount of *q* items, received from a non-cooperative or unknown buyer, having the price lower than

*q*minPrice*, when the negotiation step is greater than *5*. The message *Price too low* is sent from the seller to the buyer;

```
(defrule reject2
 (declare (salience 50))
 (NegotiationObject {buyerClassification == "nc" ||
buyerClassification == "u"} {step > 5}
                    (currentOffer ?o) (quantity ?q))
 (Product (minPrice ?minPrice))
 (test (<= ?o (* ?q ?minPrice)))
 => (add (new Offer "reject2" ?*reject* "Price too low")))
```

3) The seller will reject the current offer *o* for a certain amount of *q* items, received from a slightly cooperative or cooperative buyer, having the price lower than *q*(minPrice+20)*, when the negotiation step is greater than *5*. The message *Price too low* is sent from the seller to the buyer;

```
(defrule reject3
 (declare (salience 50))
 (NegotiationObject {buyerClassification == "sc" ||
buyerClassification == "c"} {step > 5}
                    (currentOffer ?o) (quantity ?q))
 (Product (minPrice ?minPrice))
 (test (<= ?o (* ?q (+ ?minPrice 20))))
 => (add (new Offer "reject3" ?*reject* "Price too low")))
```

4) The seller will reject the current offer *o* for a certain amount of *q* items, received from a very cooperative or highly cooperative buyer, having the price lower than *q*(minPrice+40)*, when the negotiation step is greater than *5*. The message *Price too low* is sent from the seller to the buyer;

```
(defrule reject4
 (declare (salience 50))
 (NegotiationObject {buyerClassification == "vc" ||
buyerClassification == "hc"} {step > 5}
                    (currentOffer ?o) (quantity ?q))
 (Product (minPrice ?minPrice))
 (test (<= ?o (* ?q (+ ?minPrice 40))))
=> (add (new Offer "reject4" ?*reject* "Price too low")))
```

For the **PROPOSE** communication primitive of the seller, there are the following rules:

1) The price proposed by a seller, in the beginning, for a certain quantity *q* of products, having the price between *minPrice* and *maxPrice*, to a non-cooperative or unknown buyer, is equal to *q*(maxPrice+20)*;

```
 (defrule firstPrice1
 (declare (salience 10))
 (NegotiationObject {step == 0} {buyerClassification == "nc" ||
buyerClassification == "u"} (quantity ?q))
 (Product (maxPrice ?maxPrice))
 => (add (new Offer "firstPrice1" (* (+ ?maxPrice 20) ?q)
?*propose*)))
```

2) The price proposed by a seller, in the beginning, for a certain quantity *q* of products, having the price between *minPrice* and *maxPrice*, to a slightly cooperative or cooperative buyer, is equal to *q*maxPrice*;

```
(defrule firstPrice2
  (declare (salience 10))
  (NegotiationObject {step == 0} {buyerClassification == "sc" ||
buyerClassification == "c"} (quantity ?q))
  (Product (maxPrice ?maxPrice) )
=> (add (new Offer "firstPrice2" (* ?maxPrice ?q) ?*propose*)))
```

3) The price proposed by a seller, in the beginning, for a certain quantity *q* of products, having the price between *minPrice* and *maxPrice*, to a very cooperative or highly cooperative buyer, is equal to *q*(maxPrice-20)*;

```
(defrule firstPrice3
  (declare (salience 10))
  (NegotiationObject {step == 0} {buyerClassification == "vc" ||
buyerClassification == "hc"} (quantity ?q))
  (Product (maxPrice ?maxPrice) )
=> (add (new Offer "firstPrice3" (* (- ?maxPrice 20) ?q) ?*propose*)))
```

The seller agent has three different strategies used during negotiation. It can employ either the linear strategy, the conceder strategy or the boulware strategy. There are different rules used for each strategy.

For the **linear strategy** of the seller, there are the following rules:

1) The seller will decrease its previous offer *o* with *2*, to the non-cooperative or unknown buyers, when the negotiation step is greater than *1*;

```
  (defrule newPrice4
  (declare (salience 10))
  (NegotiationObject {step > 1} (previousOffer ?o) {buyerClassification
== "nc" || buyerClassification == "u"}) (Strategy {strategy == ?*linear*})
   => (add (new Offer "newPrice4" (- ?o 2) ?*propose*)))
```

2) The seller will decrease its previous offer *o* with *4*, to the slightly cooperative or cooperative buyers, when the negotiation step is greater than *1*;

```
(defrule newPrice5
  (declare (salience 10))
  (NegotiationObject {step > 1} (previousOffer ?o) {buyerClassification
== "sc" || buyerClassification == "c"}) (Strategy {strategy == ?*linear*}
  => (add (new Offer "newPrice5" (- ?o 4) ?*propose*)))
```

3) The seller will decrease its previous offer *o* with *6*, to the very cooperative or highly cooperative buyers, when the negotiation step is greater than *1*;

```
(defrule newPrice6
  (declare (salience 10))
  (NegotiationObject {step > 1} (previousOffer ?o) {buyerClassification
== "vc" || buyerClassification == "hc"}) (Strategy {strategy == ?*linear*})
  => (add (new Offer "newPrice6" (- ?o 6) ?*propose*)))
```

For the **conceder strategy** of the seller, there are the following rules:

1) The seller will decrease its previous offer *o* with *25*, to the non-cooperative or unknown buyers, when the negotiation step is greater than *1* and lower than *6*;

```
  (defrule newPrice7
```

```
      (declare (salience 10))
      (NegotiationObject {step > 1 && step < 6} (previousOffer ?o)
{buyerClassification == "nc" || buyerClassification == "u"})
      (Strategy {strategy == ?*conceder*})
      => (add (new Offer "newPrice7" (- ?o 25) ?*propose*)))
```

2) The seller will decrease its previous offer *o* with *35*, to the slightly cooperative or cooperative buyers, when the negotiation step is greater than *1* and lower than *6*;

```
     (defrule newPrice8
      (declare (salience 10))
      (NegotiationObject {step > 1 && step < 6} (previousOffer ?o)
{buyerClassification == "sc" || buyerClassification == "c"})
      (Strategy {strategy == ?*conceder*})
    =>(add (new Offer "newPrice7" (- ?o 35) ?*propose*)))
```

3) The seller will decrease its previous offer *o* with *45*, to the very cooperative or highly cooperative buyers, when the negotiation step is greater than *1* and lower than *6*;

```
     (defrule newPrice9
      (declare (salience 10))
      (NegotiationObject {step > 1 && step < 6} (previousOffer ?o)
{buyerClassification == "vc" || buyerClassification == "hc"})
      (Strategy {strategy == ?*conceder*})
      => (add (new Offer "newPrice7" (- ?o 45) ?*propose*)))
```

4) The seller will decrease its previous offer *o* with *5*, to the non-cooperative or unknown buyers, when the negotiation step is greater than *6*;

```
     (defrule newPrice10
      (declare (salience 10))
      (NegotiationObject {step >= 6} (previousOffer ?o)
{buyerClassification == "nc" || buyerClassification == "u"}) (Strategy
{strategy == ?*conceder*})
      => (add (new Offer "newPrice10" (- ?o 5) ?*propose*)))
```

5) The seller will decrease its previous offer *o* with *10*, to the slightly cooperative or cooperative buyers, when the negotiation step is greater than *6*;

```
     (defrule newPrice11
      (declare (salience 10))
      (NegotiationObject {step >= 6} (previousOffer ?o)
{buyerClassification == "sc" || buyerClassification == "c"}) (Strategy
{strategy == ?*conceder*})
    => (add (new Offer "newPrice11" (- ?o 10) ?*propose*)))
```

6) The seller will decrease its previous offer *o* with *15*, to the very cooperative or highly cooperative buyers, when the negotiation step is greater than *6*;

```
     (defrule newPrice12
      (declare (salience 10))
      (NegotiationObject {step >= 6} (previousOffer ?o)
{buyerClassification == "vc" || buyerClassification == "hc"}) (Strategy
{strategy == ?*conceder*})
    => (add (new Offer "newPrice12" (- ?o 15) ?*propose*)))
```

For the **boulware strategy** of the seller, there are the following rules:

1) The seller will decrease its previous offer *o* with *0.1*, when the negotiation step is greater than *1* and the time elapsed in the negotiation is less than *4* seconds, represented by the global variable *boulwareTime1*;

```
      (defrule newPrice13
      (declare (salience 10))
      (NegotiationObject {step > 1} (previousOffer ?o) {timeElapsed <
?*boulwareTime1*})
      (Strategy {strategy == ?*boulware*})
    => (add (new Offer "newPrice13" (- ?o 0.1) ?*propose*)))
```

2) The seller will decrease its previous offer *o* with *10*, when the negotiation step is greater than *1* and the time elapsed in the negotiation is greater than *4* seconds, represented by the global variable *boulwareTime1*;

```
     (defrule newPrice14
      (declare (salience 10))
      (NegotiationObject {step > 1} (previousOffer ?o) {timeElapsed >=
?*boulwareTime1*})
      (Strategy {strategy == ?*boulware*})
      => (add (new Offer "newPrice14" (- ?o 10) ?*propose*)))
```

3) The seller will decrease its previous offer *o* with *20*, when the negotiation step is greater than *1* and the time elapsed in the negotiation is greater than *4.5* seconds, represented by the global variable *boulwareTime2*;

```
     (defrule newPrice15
      (declare (salience 10))
      (NegotiationObject {step > 1} (previousOffer ?o) {timeElapsed >=
?*boulwareTime2*})
      (Strategy {strategy == ?*boulware*})
      => (add (new Offer "newPrice14" (- ?o 20) ?*propose*)))
```

8.2. One-to-One Automated Negotiation Scenario

The following graphics are obtained using the data collected from the statistics file, generated after the multi-agent system runs the automated negotiation between agents. When all the buyers finish their purchases, all the negotiation information is recorded in the statistics file.

In the scenario, the buyer wants to rent 36 rooms with 12 different characteristics, 3 rooms of each type. The seller has a total of 120 rooms, 10 rooms for each type.

A snapshot of the Sniffer Agent from the Jade environment, representing the exchange of messages in a one-to-one negotiation, is presented in Figure 8.1.

In Figure 8.2 is described a one-to-one negotiation, in which is plotted the gain versus the number of negotiation rounds. The graphics obtained show that, when using the same negotiation strategy, the buyer and seller gain are almost the same. They can obtain different gains if they use distinct negotiation strategies.

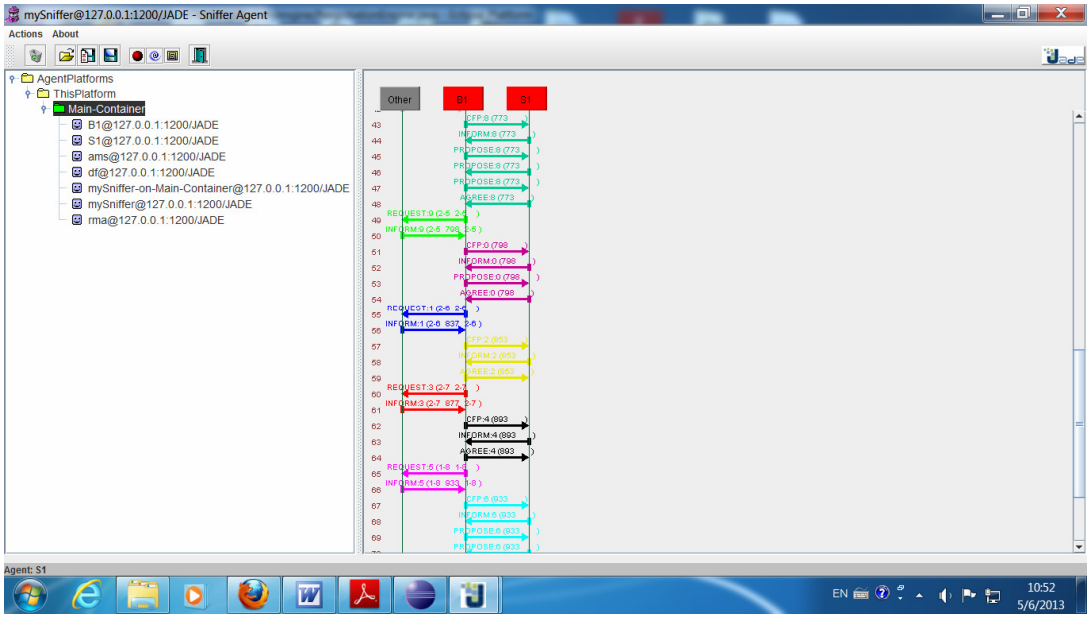


Figure 8.1. Messages Exchange Captured with the Sniffer Agent for a One-to-One Negotiation

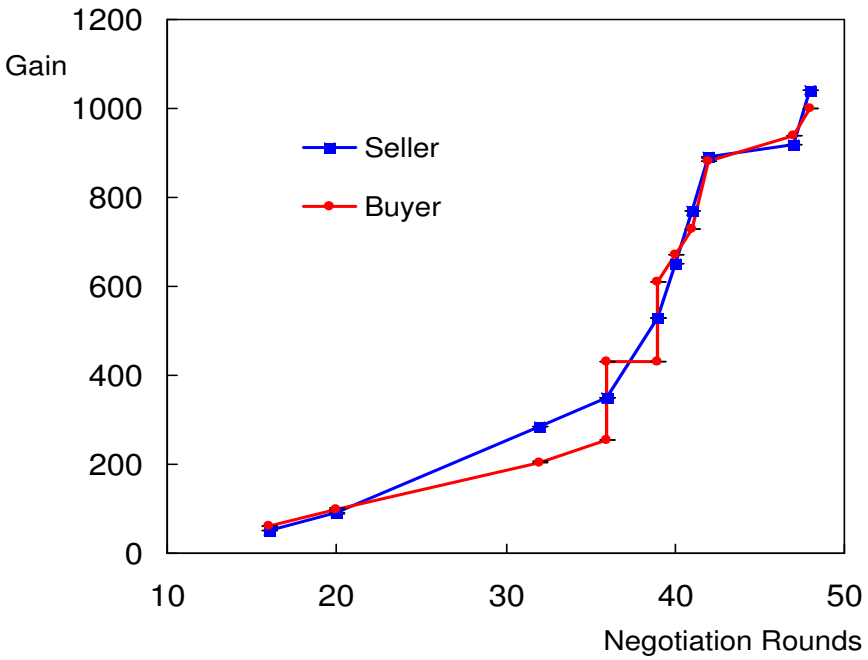


Figure 8.2. Transactional Gain Dependence of Negotiation Rounds Number for a One-to-One Negotiation

The gain obtained for each cooperation class, with respect to the number of negotiations, for the seller and respectively the buyer, are represented in Figures 8.3 and 8.4.

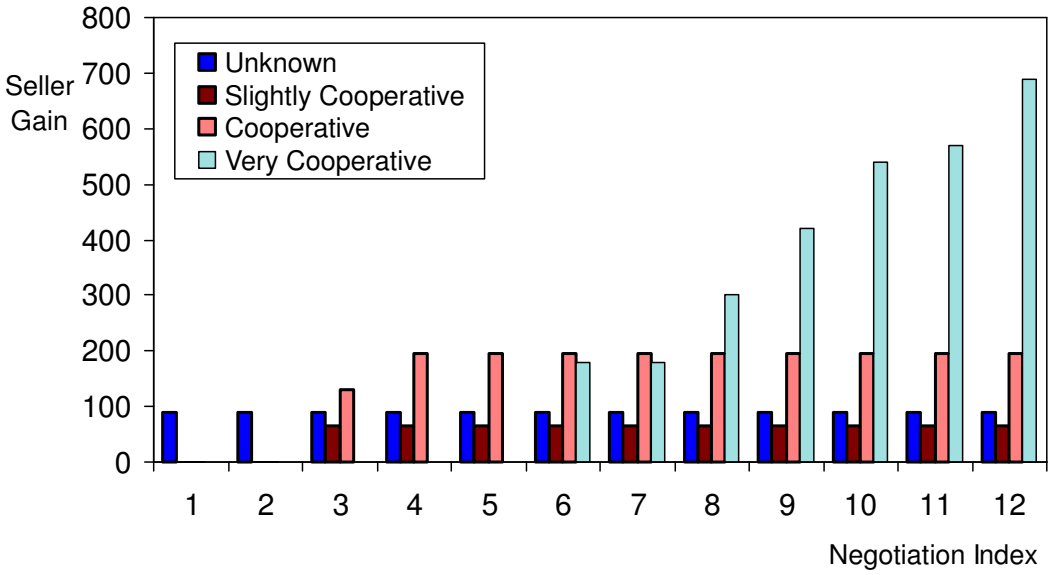


Figure 8.3. Transactional Gain in each Cooperation Class for each Negotiation Index for the Seller in a One-to-One Negotiation

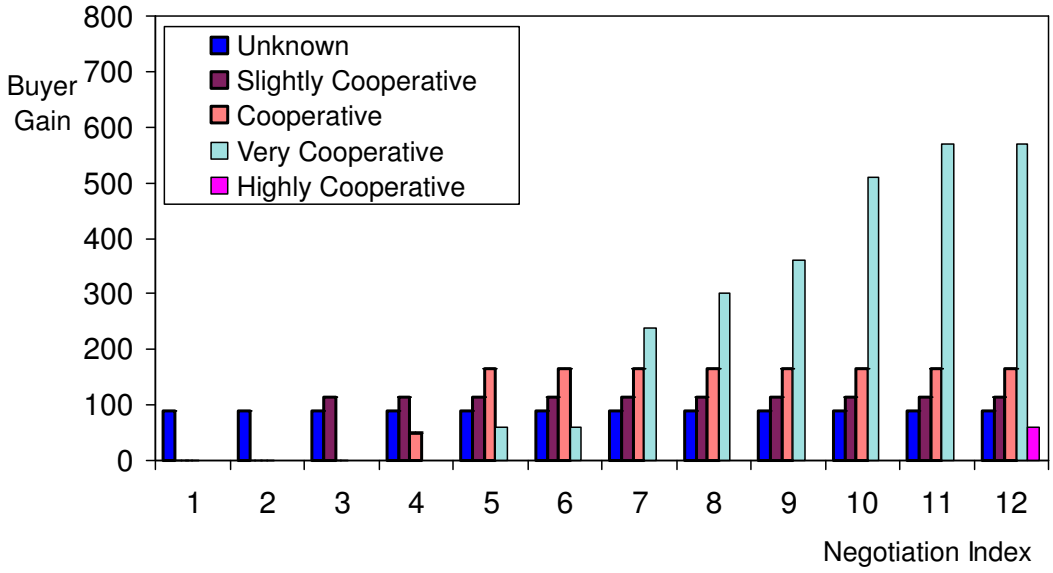


Figure 8.4. Transactional Gain in each Cooperation Class for each Negotiation Index for the Buyer in a One-to-One Negotiation

From the Figures 8.3 and 8.4 it can be deduced that, during negotiation, the cooperation classes of the agents are changed, as more negotiation rounds are performed. In the first negotiations, both agents belong to the unknown cooperation class. As they come to know more about each other, they change the classification of the cooperation potential of the partner agent. In the last negotiation, the buyer becomes highly cooperative, while the seller remains very cooperative.

The gain for the buyer and for the seller versus the supply/demand ratio is represented in Figure 8.5.

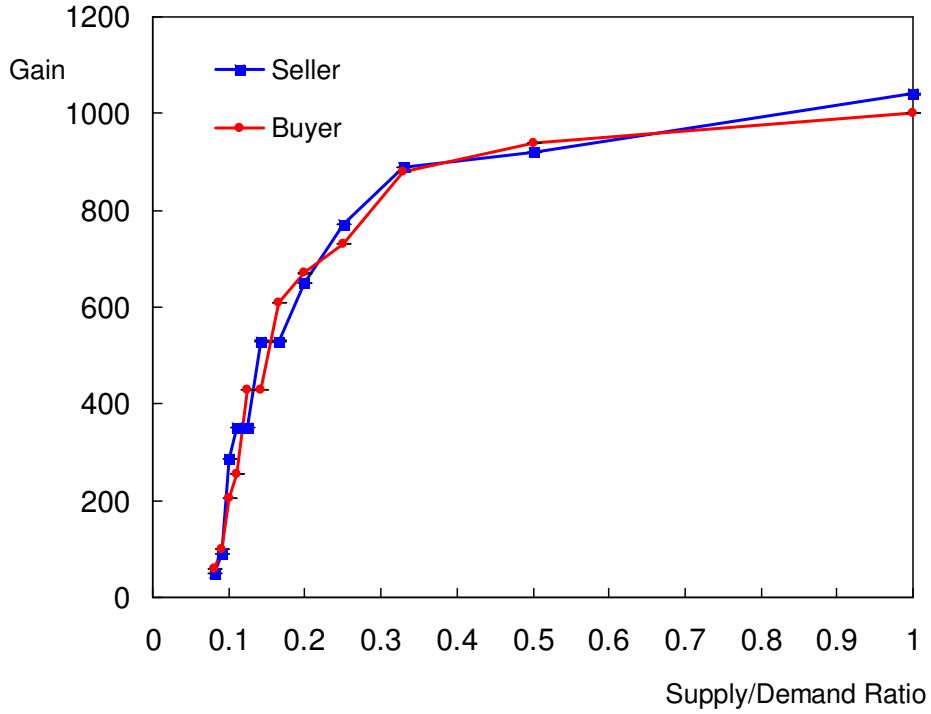


Figure 8.5. Transactional Gain Dependence of Supply/Demand Ratio for a One-to-One Negotiation

8.3. Two-to-One Automated Negotiation Scenario

In the next scenario, two buyers want to rent a total of 100 rooms. The first buyer wants to rent 40 rooms with 10 different characteristics, 4 rooms of each type and the second buyer 60 rooms with 10 different characteristics, 6 rooms of each type. The seller has a total of 100 rooms, 10 rooms for each type.

A snapshot of the Sniffer Agent from the Jade environment, representing the exchange of messages in a two-to-one negotiation, is presented in Figure 8.6.

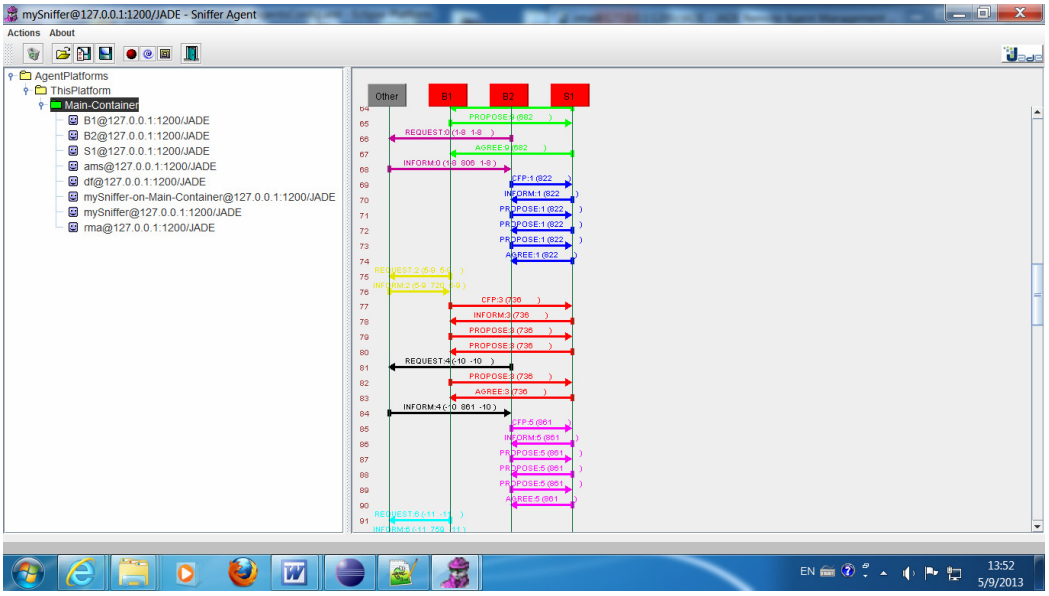


Figure 8.6. Messages Exchange Captured with the Sniffer Agent for a Two-to-One Negotiation

In Figure 8.7 is described a two-to-one negotiation, in which is plotted the gain versus the number of negotiation rounds.

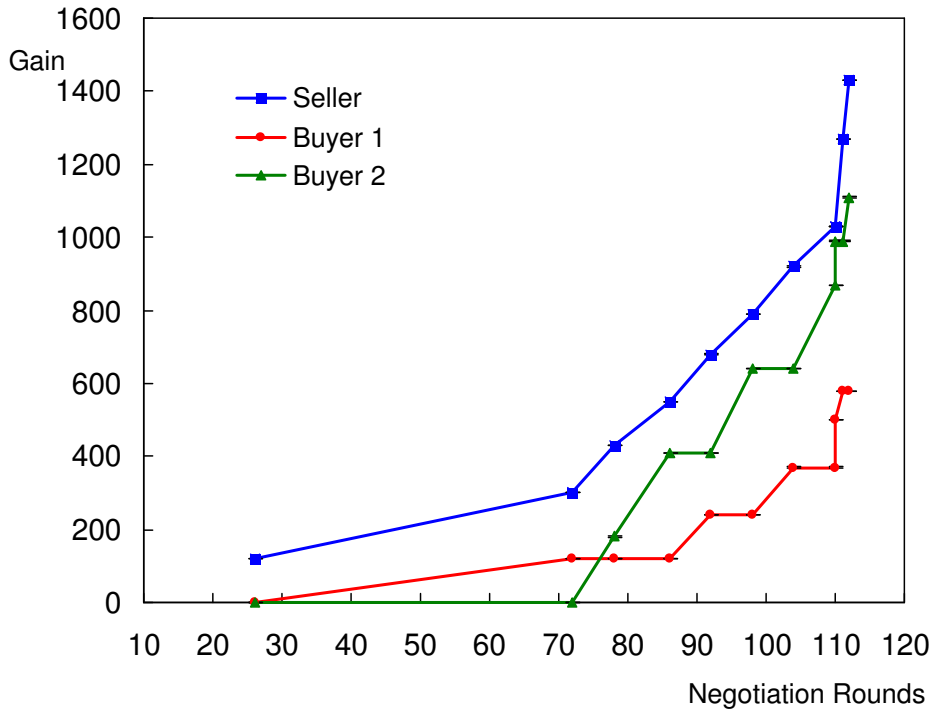


Figure 8.7. Transactional Gain Dependence of Negotiation Rounds Number for a Two-to-One Negotiation

The classification of the cooperation classes with respect to the number of negotiation for the seller and respectively the buyers are represented in Figures 8.8, 8.9, and 8.10.

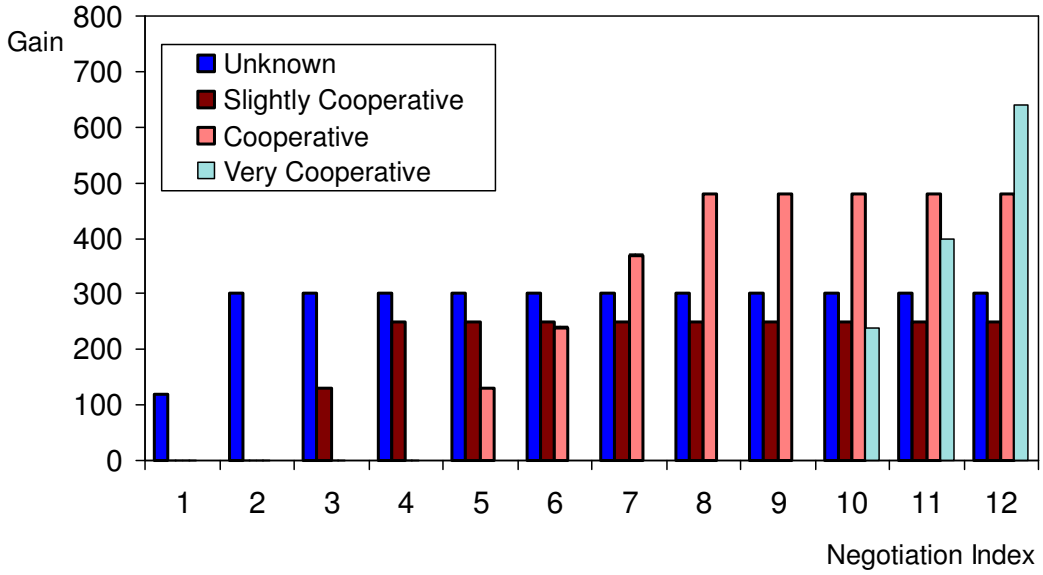


Figure 8.8. Transactional Gain in each Cooperation Class for each Negotiation Index for the Seller in a Two-to-One Negotiation

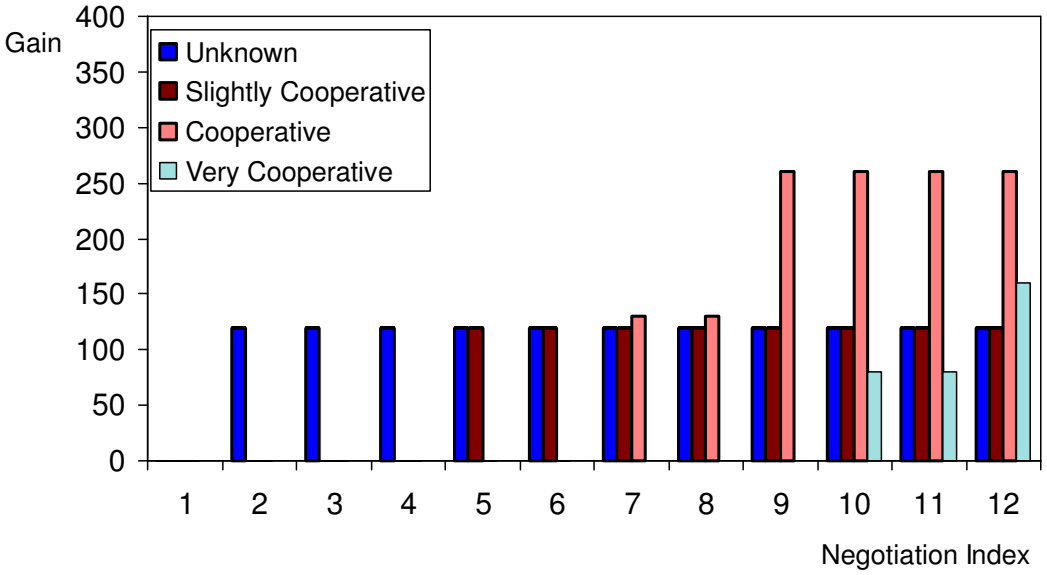


Figure 8.9. Transactional Gain in each Cooperation Class for each Negotiation Index for the First Buyer in a Two-to-One Negotiation

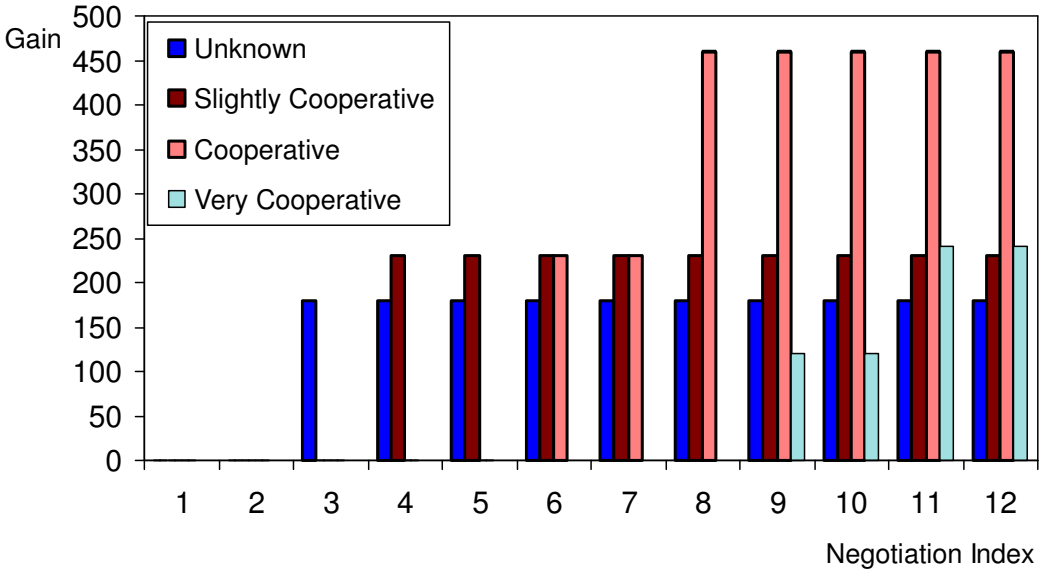


Figure 8.10. Transactional Gain in each Cooperation Class for each Negotiation Index for the Second Buyer in a Two-to-One Negotiation

The number of negotiation rounds for each cooperation class, in the case of a two-to-one negotiation, for the seller and for the buyers, are represented in Figures 8.11, 8.12, and 8.13.

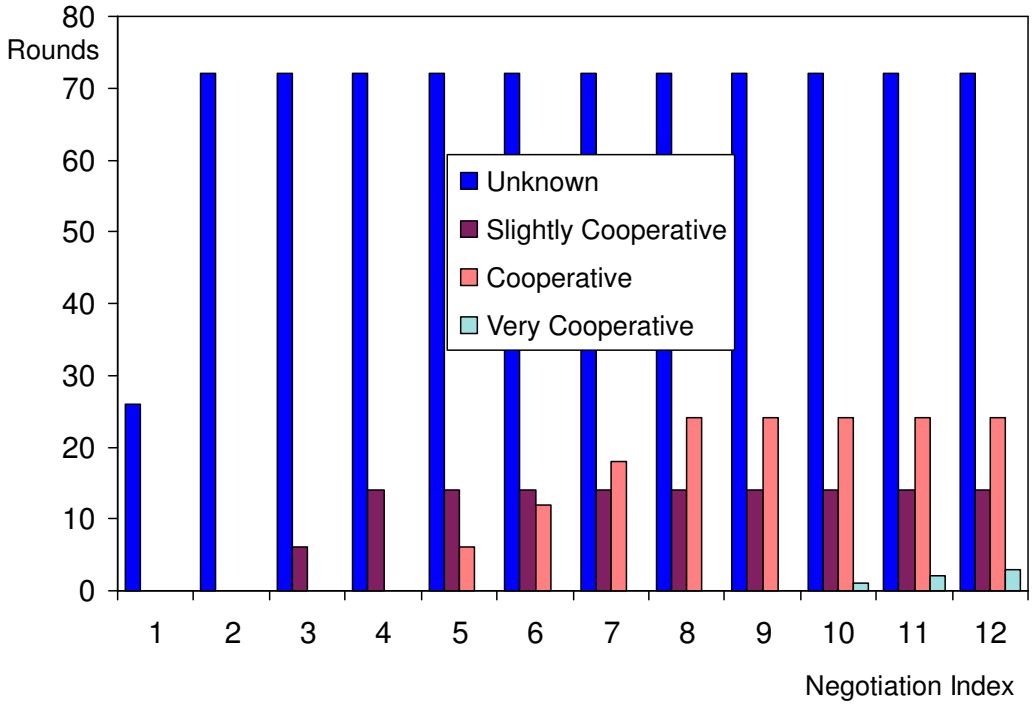


Figure 8.11. The Number of Negotiation Rounds for each Cooperation Class, in the Case of a Two-to-One Negotiation, for the Seller

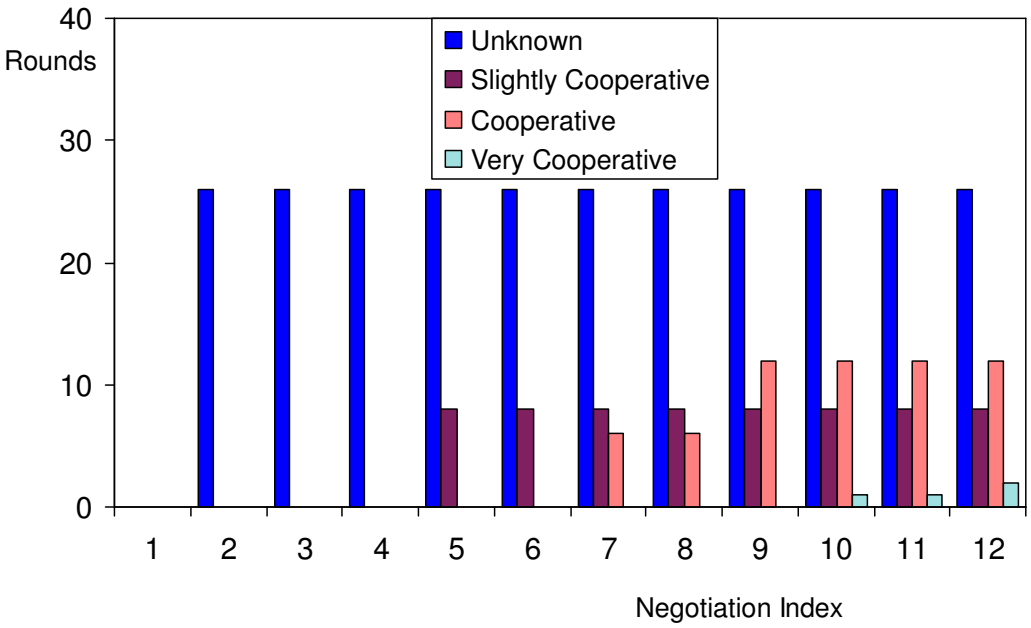


Figure 8.12. The Number of Negotiation Rounds for each Cooperation Class, in the Case of a Two-to-One Negotiation, for the First Buyer

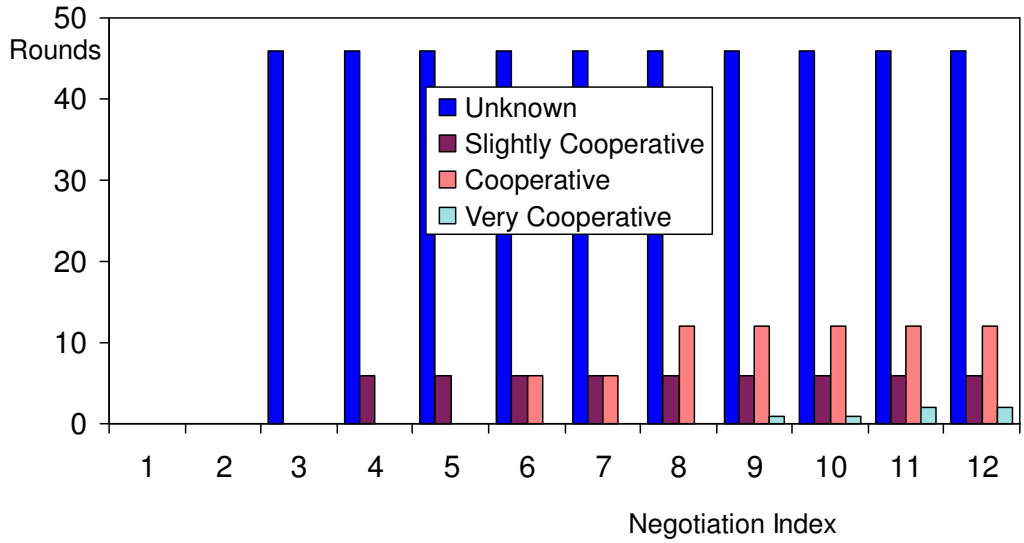


Figure 8.13. The Number of Negotiation Rounds for each Cooperation Class, in the Case of a Two-to-One Negotiation, for the Second Buyer

While the number of negotiation rounds is increasing, the classification in cooperation classes of the partner agent is performed. When the negotiation ends, the partner agents become very cooperative.

8.4. Many-to-One Automated Negotiation Scenario

The next B2B travel agency scenario involves one seller and a different number of buyers, from 1 to 10. Ten different negotiations are performed, the number of buyers being successively increased by one. The seller has 200 hotel rooms to rent, of 4 types. Each buyer wants to rent 20 rooms, 5 rooms of each different type. The seller asks for the rooms' prices between 70 and 100 monetary units. The Figure 8.14 displays the total gain of the seller with respect to the number of buyer agents acting in the negotiation process. The seller gain is increasing linearly, when there are up to 6 buyer agents. Then, its gain is increasing exponentially, when there are more than 6 buyer agents in the virtual market.

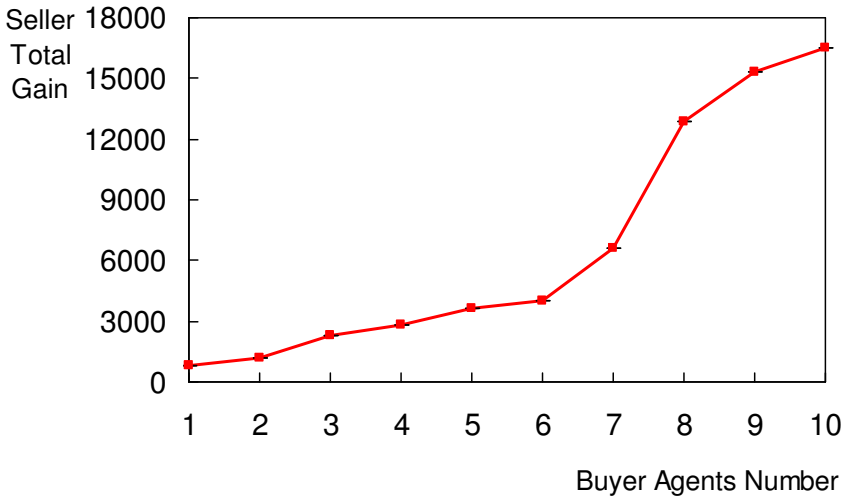


Figure 8.14. The Seller Total Gain with Respect to the Number of Buyer Agents Acting in the Negotiation

The number of negotiation rounds with respect to the number of buyer agents acting in the negotiation is represented in the Figure 8.15.

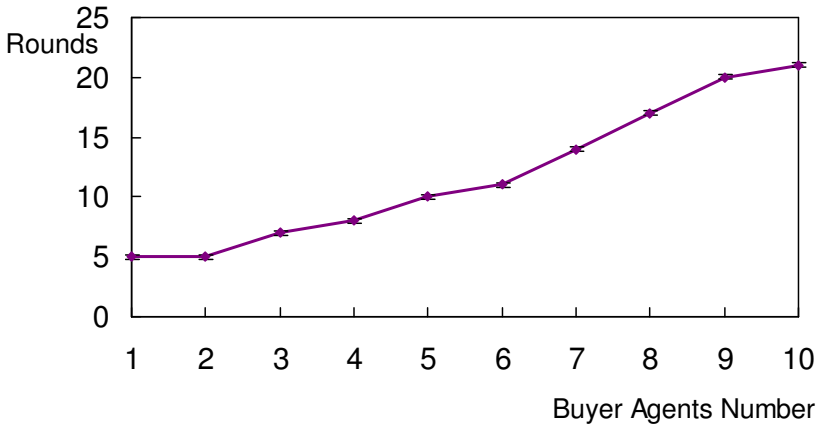


Figure 8.15. The Number of Negotiation Rounds with Respect to the Number of Buyer Agents Acting in the Negotiation

Chapter 9. Business Models Use Cases for Automated Negotiation

After presenting the automated negotiation business model with a travel agency in Chapter 8, the current chapter develops three other business models use cases, each of them having some particularities, described in the beginning of each section.

9.1. Real Estate Agency Automated Negotiation Business Model

The following scenario develops an automated negotiation between real estate agencies and the real estate developers. The houses to be sold have different attributes, expressed in the **XML configuration file**, associated to this business model. The negotiation is done based on price, but there are also other attributes in the configuration file, such as the number of rooms, rooms' dimensions, finishing quality, location of the house and others, displayed in Table 9.1.

Table 9.1. Negotiation Requirements and Their Priorities

Buyer Agent Requirements	Priority
Buy a House / an Apartment	10
City / Area of the City	10
No of floors / No of rooms	10
Land Surface / House Surface / Apartment Surface	10
No of bathrooms / kitchens / balcony	9
With / without furniture	7
Finishing quality	9
Close to supermarkets / shopping area	8
Close to parks	9
Close to public transportation	10
Close to kindergarten / school / high school / university	8
New building / old building / refurbished building	10
With / without parking area	7

As **a difference to the rules associated to the travel agency negotiation scenario**, in the **real estate agency business model**, the **REJECT rules based on price are eliminated**. **The negotiation is ended** either **when the time expires,** or **when one of the agents sends an ACCEPT message.**

The rules for the real estate agency are divided into rules for the buyer and for the seller agents. Further on, there are different rules associated to each communication primitive of the agents, and also for each strategy of the agent. These rules are described in details in Annex 1.

9.1.1. Three-to-One Automated Negotiation Scenario

The following B2B scenario involves a real estate agency. There are three buyers, wanting to buy houses from the real estate agent, each buyer having a different negotiation strategy: 1-linear, 2-boulware, and 3-conceder. The requirements of the agents are represented in Table 9.2.

Table 9.2. The Buyer Agents' Requirements and the Seller Offer

Agent	House Type	Quantity	Price
B_1, B_2, B_3	2 Rooms	5	40000-60000
B_1, B_2, B_3	3 Rooms	5	80000-100000
S_1	2 Rooms	20	45000-70000
S_1	3 Rooms	20	85000-110000

The results are presented in Figure 9.1. The buyers' gain obtained after the simulation certifies the theoretical behavior. The higher gain is obtained for the buyer using the conceder strategy, the medium gain corresponds to the linear strategy and the smaller gain for the boulware strategy.

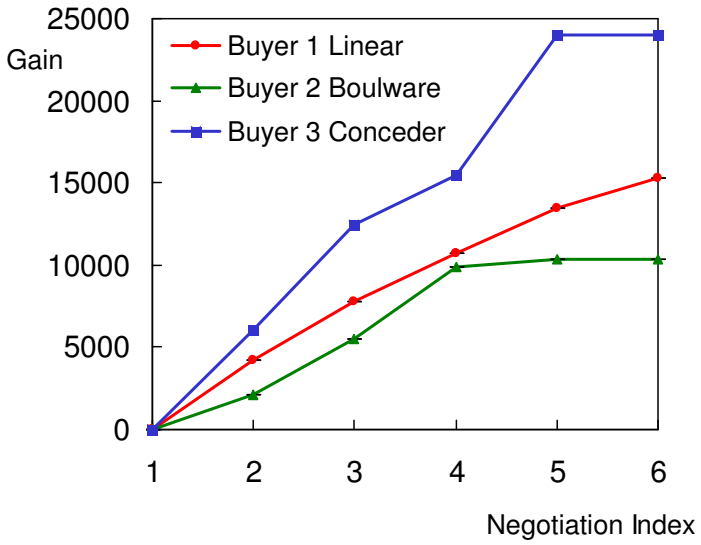


Figure 9.1. The Three Buyers Gain in a Negotiation Process with Six Steps, each using a Different Negotiation Strategy, in a B2B Scenario involving a Real Estate Agency

The Figure 9.2 shows the buyers gain, divided for each cooperation class (unknown, non cooperative, slightly cooperative), obtained after the negotiation is performed. The data regarding the gain is read from the statistics file generated by the program.

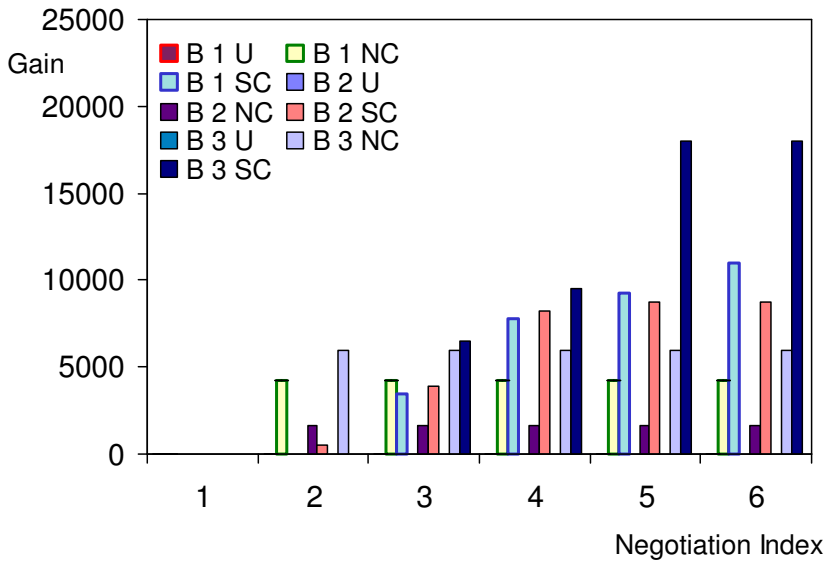


Figure 9.2. The Three Buyers Gain for each Cooperation Class, where U – Unknown, NC – Non-cooperative, and SC – Slightly cooperative

Figure 9.3 shows the seller gain, divided for each cooperation class (unknown, non cooperative, slightly cooperative), obtained after the negotiation is performed.

Figure 9.4 represents the gain of the three types of buyers (1-linear, 2-boulware, and 3-conceder) versus the gain of the seller.

Figure 9.5 displays each buyer weighted gain, for each step of the negotiation (the negotiation steps are displayed in a clockwise direction).

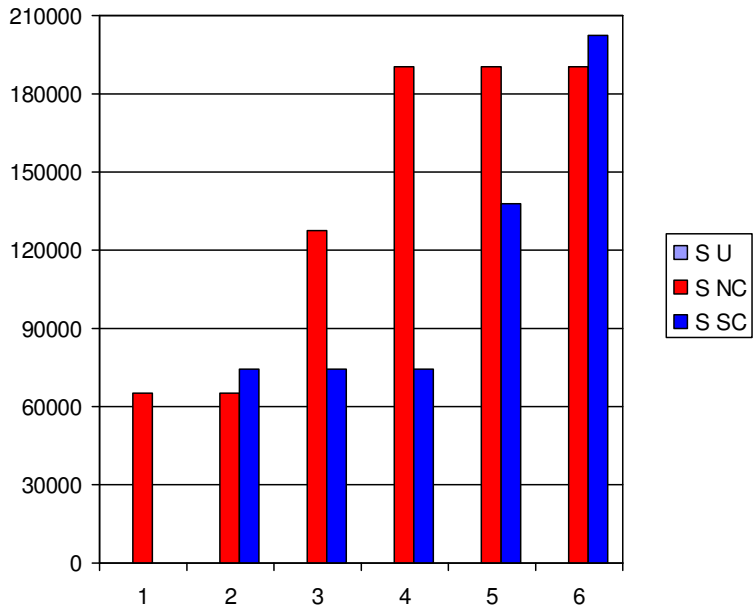


Figure 9.3. The Seller Gain for Each Cooperation Class in the Negotiation Process with Six Steps

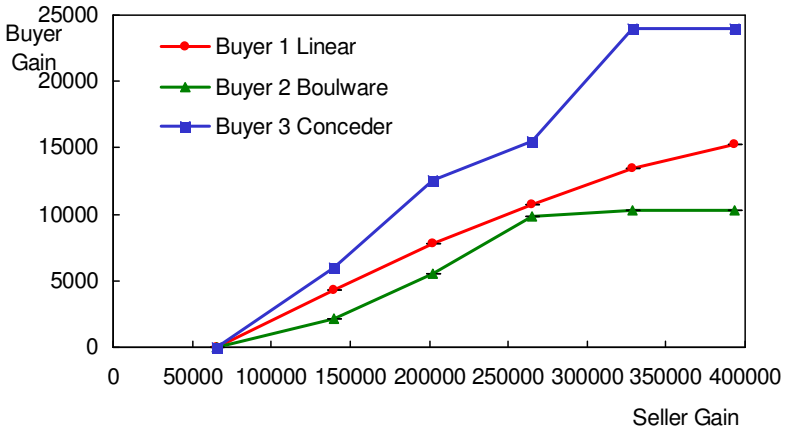
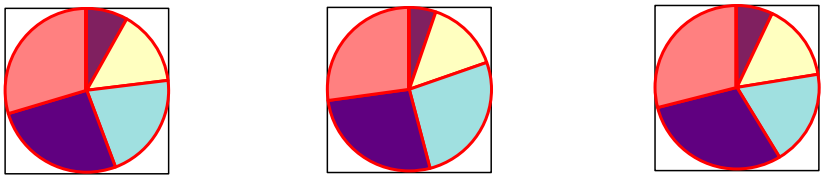


Figure 9.4. The Buyers' Gains, using Different Types of Strategies, versus the Seller Gain



B_1 linear B_2 boulware B_3 conceder

Figure 9.5. The Buyers Weighted Gain during the Negotiation Rounds

For the buyer using the linear strategy, the gain increases linearly with the negotiation index. In the first negotiation, there is no gain for all the buyers. The weighted gain in the third negotiation round is almost the same for all the three buyers. In the fourth negotiation, the buyer using a boulware strategy gets a higher weighted gain than the other two buyers. In the last negotiation, the buyer employing the boulware strategy obtains a lower weighted gain than the other two buyers.

9.1.2. Two-to-Three Automated Negotiation Scenario

The following scenario involves two buyers and three sellers. The buyers use a linear strategy, but the sellers are using different strategies: S_1 has a linear strategy, S_2 a conceder strategy, and S_3 a boulware strategy.

Figure 9.6 shows a snapshot of the negotiation system, in which the agents use different negotiation strategies.

Figure 9.7 displays the buyers gain versus the negotiation index.

Figure 9.8 represents the sellers gain versus the negotiation index. Each seller uses a different strategy during negotiation.

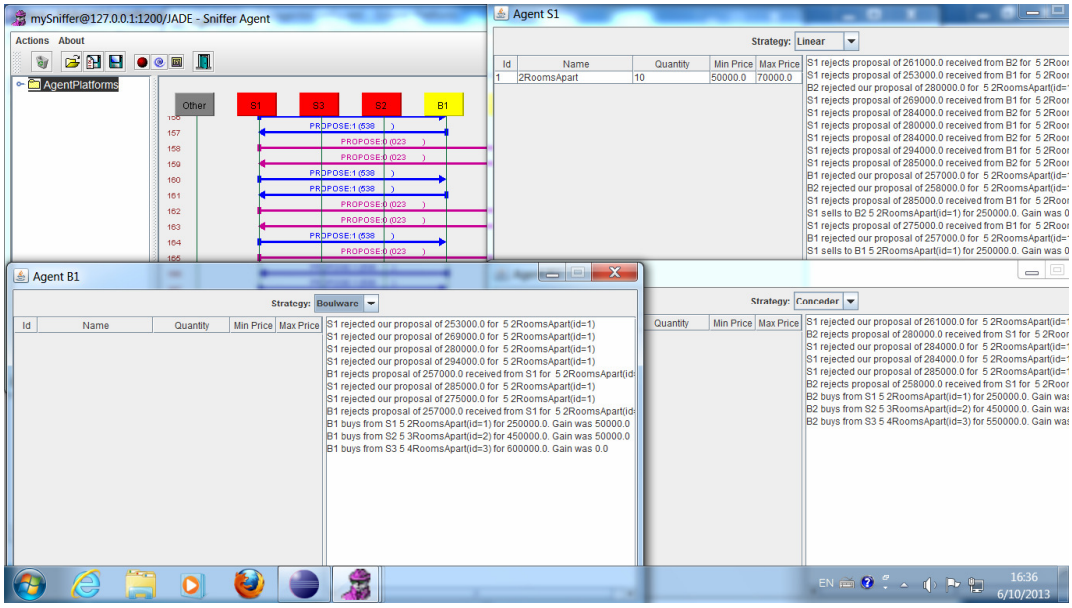


Figure 9.6. Screen Capture Showing the System in Action

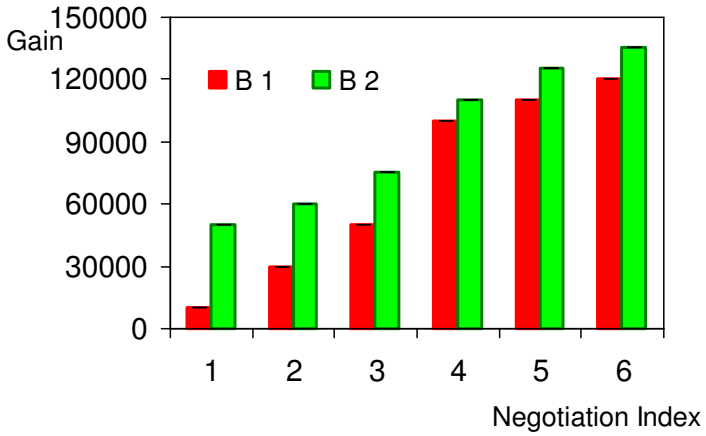


Figure 9.7. The Buyers' Gain with respect to the Negotiation Index

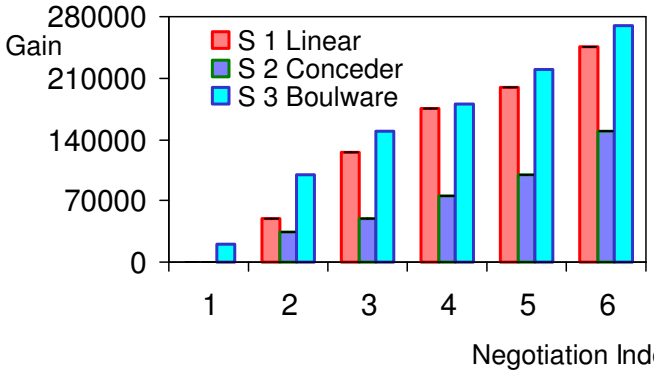


Figure 9.8. The Sellers' Gain with respect to the Negotiation Index

The sellers' gain obtained after the simulation certifies the theoretical behavior. The higher gain is obtained for the seller using the boulware strategy, the medium gain corresponds to the linear strategy and the smaller gain is obtained for the conceder strategy.

9.2. Car Dealer Automated Negotiation Business Model

This scenario involves car dealers negotiating with car factories, in order to buy sets of cars. Each type of car has different characteristics, expressed in the **XML configuration file** associated to each agent participating in the negotiation.

As **a difference to the previous business models**, there is **added a seller discount for the first price, depending on the required quantity of cars** (as the number of cars to be bought is higher, the discount increases). Table 9.4 displays the main characteristics of the car and their priorities for the dealers.

The rules for the car dealer business model are divided into rules for the buyer and for the seller agents. Further on, there are different rules associated to each communication primitive of the agents, and also for each strategy of the agent. These rules are described in details in Annex 2.

Table 9.3. Negotiation Characteristics and Their Priorities

Buyer Agent Characteristics	Priority
Producer	10
Price	8
Maximum Speed	5
Time to Maximum Speed	3
Number of Seats	9
Number of Airbags	7
Trunk Size	6
Car Type	9
Car Fuel	8
Start Stop System	3
Audio System	8
Video System	2

9.2.1. One-to-Five Automated Negotiation Scenario

The next scenario involves one buyer and five sellers. The buyer wants 10 cars with 5 different characteristics. Each seller has a certain type of car. The car prices increase from S_1 to S_5, because the car quality increases from S_1 to S_5. Figure 9.9 displays a screen capture of the system in action.

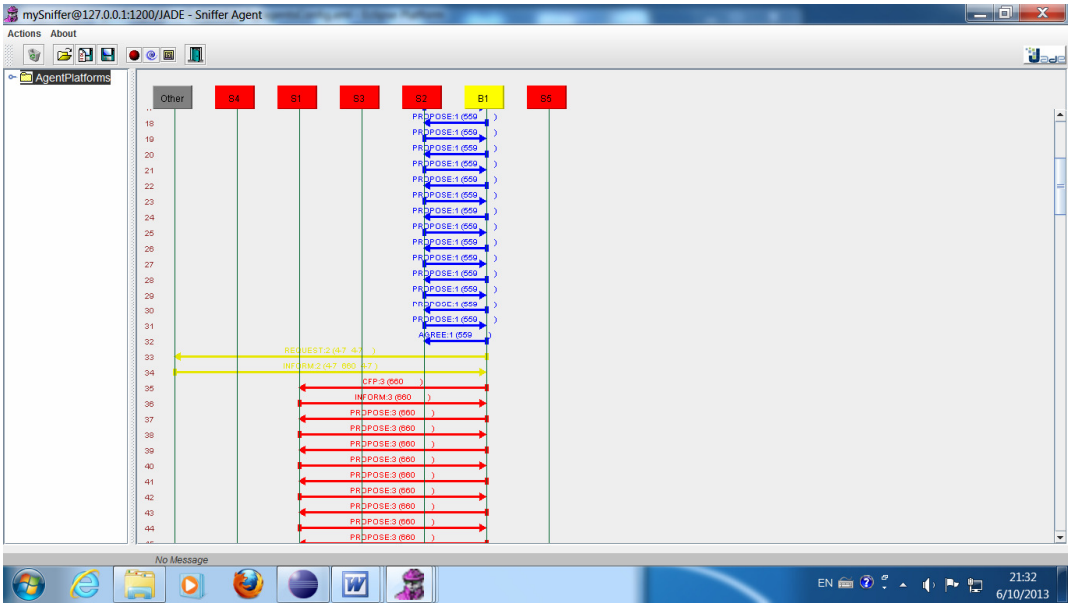


Figure 9.9. Snapshot of the Running System

Figure 9.10 displays the sellers' weighted gain obtained after negotiation (the sellers are displayed in a clockwise direction). The sellers gain increases depending on the car quality. As the price of the car increases and the car is a top one, the seller gain has higher values.

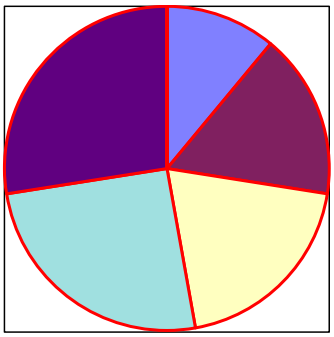


Figure 9.10. Sellers Weighted Gain

9.2.2. Three-to-Three Automated Negotiation Scenario

The following B2B scenario involves 3 car factory producers and 3 car dealers. Each dealer wants to buy 2 car types, 15 cars of each type. The sellers are using different strategies during negotiation: the first one (S_1) uses a linear strategy (L), the second (S_2) employs a conceder strategy (C), and the third one (S_3) a boulware strategy (B).

To compare the behavior of the sellers during negotiation, the minimal and maximal prices are the same for all the sellers, for each type of car.

Figure 9.11 represents a screen capture showing the system in action.

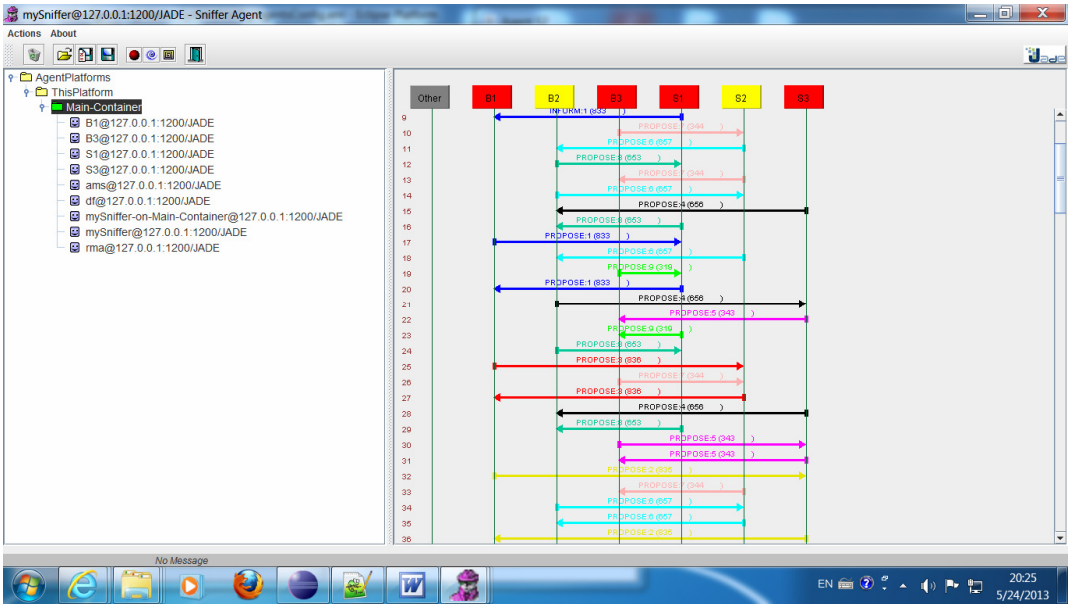


Figure 9.11. Screen Capture Showing the Multi-Agent System in Action

Figure 9.12 displays the sellers' gain with respect to the negotiation index, using different negotiation strategies: S_1 uses linear strategy, S_2 has conceder strategy, and S_3 employs boulware strategy.

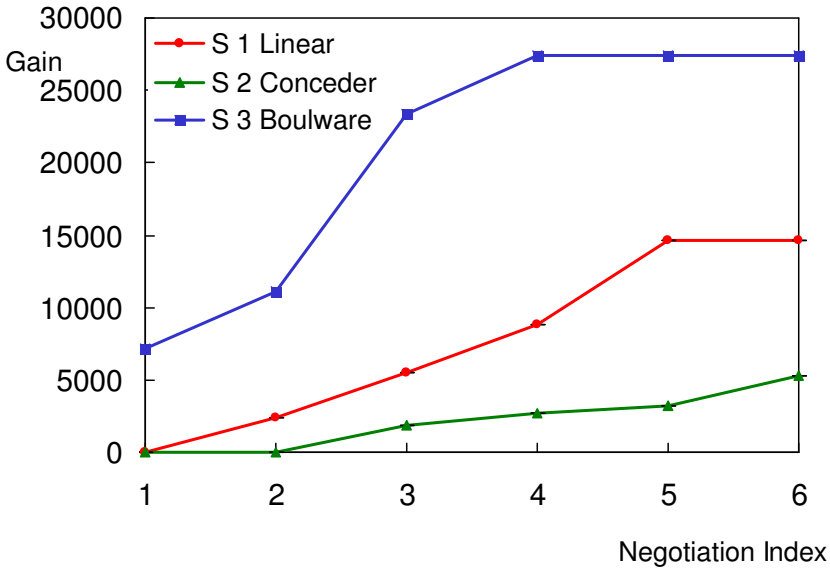
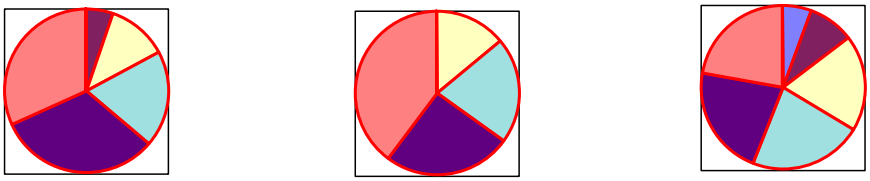


Figure 9.12. The Three Sellers Gain in a Negotiation Process with Six Steps, each using a Different Negotiation Strategy, in a B2B Scenario involving a Car Dealer

After the negotiation, it can be concluded that the higher gain is obtained by the seller using the boulware strategy, then by the one using the linear strategy, and the smaller gain is obtained by the seller using the conceder strategy.

Figure 9.13 represents each seller weighted gain, for each negotiation index (the negotiation steps are displayed in a clockwise direction).



S_1 linear S_2 conceder S_3 boulware

Figure 9.13. Weighted Gain for the Three Types of Sellers, during the Negotiation Rounds

For the seller using the linear strategy, the gain increases linearly with the negotiation index. The seller using the conceder strategy has no gain in the first two negotiations. The seller using the boulware strategy has a gain from the first negotiation round, while the other two sellers have no gain in the first round. In the third negotiation, the seller using a boulware strategy gets a higher weighted gain than the other two sellers. The weighted gain in the fourth negotiation is almost the same for all the three sellers. In the last negotiation, the seller employing the conceder strategy obtains the highest weighted gain from all the sellers.

9.3. Emergency Hospital Automated Negotiation Business Model

This scenario refers to a B2B negotiation, in a many-to-many setting, in which agents representing hospitals negotiate with different pharmaceutical companies, in order to buy different medicines and other medical equipment necessary in the hospital. The framework is inspired from the research reported in [Serbanati L.D. and Radu S., 2013].

The scenario involves buyer agents representing emergency hospitals, which want to buy rapidly medicines and the most important for them is the time in which the negotiation is concluded, while the price is not so important during negotiation.

The **negotiation time is decreased with respect to the previous business models** and now is **set to 3 seconds**. The **ACCEPT rules of the buyer are modified with respect to the previous business models**, such that **the buyer accepts different prices, depending on the negotiation time elapsed**. As **the negotiation deadline approaches, the buyer agent accepts a higher price**.

The rules for the emergency hospital business model are divided into rules for the buyer and for the seller agents. Further on, there are different rules associated to

each communication primitive of the agents, and also for each strategy of the agent. These rules are described in details in Annex 3.

9.3.1. Two-to-Two Automated Negotiation Scenario

In two laboratories of a hospital it is necessary to buy medical devices. The negotiation scenario is performed with two medical equipment sellers. Figure 9.14 shows the buyers and sellers gain versus the negotiation index.

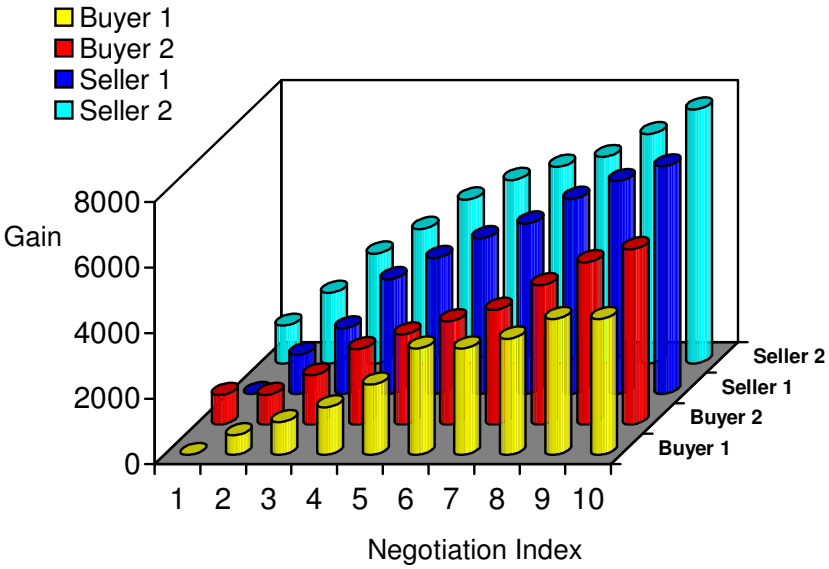


Figure 9.14. The Buyers and Sellers Gain versus the Negotiation Index

Figures 9.15 and 9.16 display the buyers gain versus the negotiation index, for each cooperation class: U - unknown, NC - non cooperative, SC - slightly cooperative, C - cooperative, VC - very cooperative, and HC - highly cooperative.

Figures 9.17 and 9.18 display the sellers gain versus the negotiation index, for each cooperation class: U - unknown, NC - non cooperative, SC - slightly cooperative, C - cooperative, VC - very cooperative, and HC - highly cooperative.

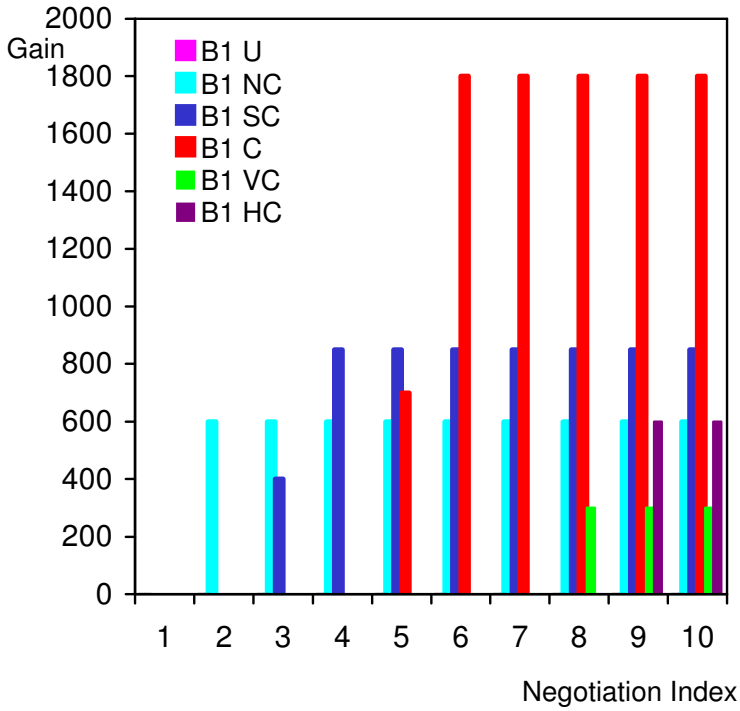


Figure 9.15. The First Buyer Gain for each Cooperation Class versus the Negotiation Index

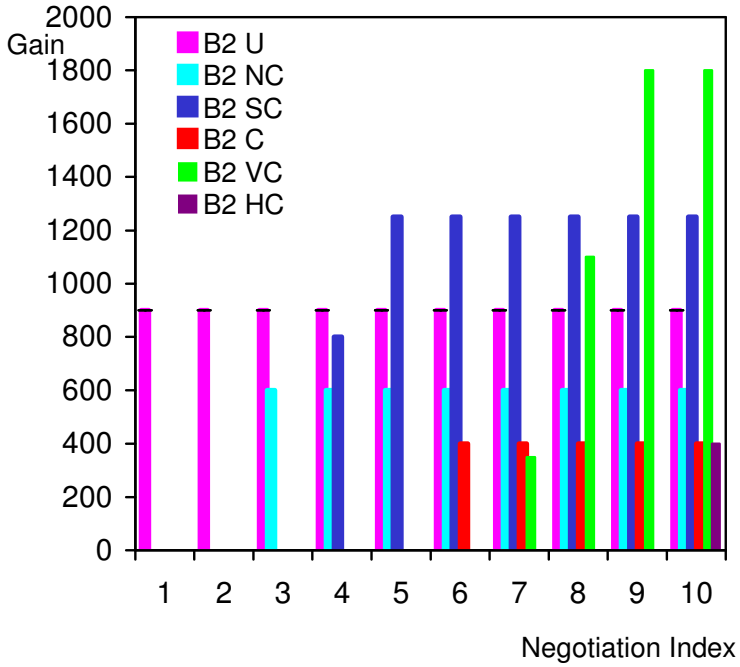


Figure 9.16. The Second Buyer Gain for each Cooperation Class versus the Negotiation Index

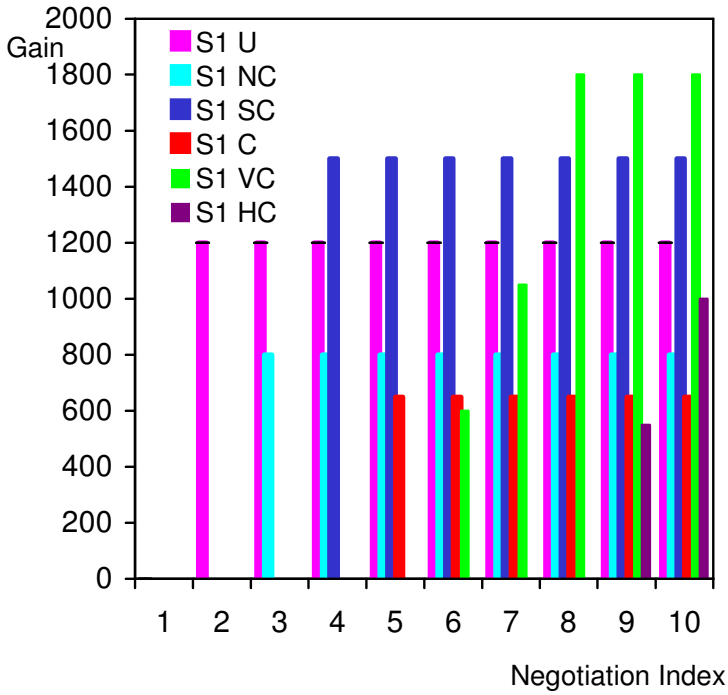


Figure 9.17. The First Seller Gain for each Cooperation Class versus the Negotiation Index

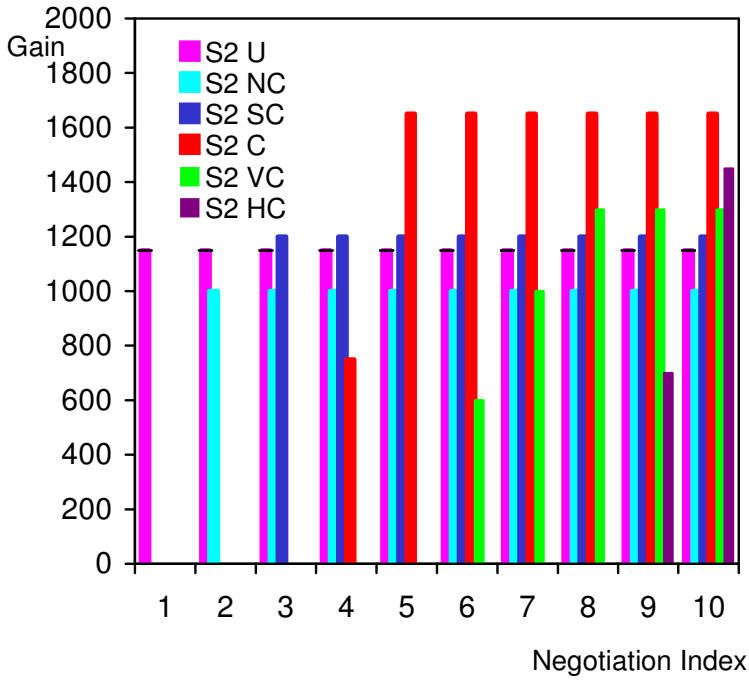


Figure 9.18. The Second Seller Gain for each Cooperation Class versus the Negotiation Index

When negotiation begins, the agents don't know each other and they are classifying the partner agent in the unknown class.While more negotiation rounds are performed, the classification of the partner agent is changed. The first buyer becomes cooperative in the fifth negotiation round, while the second buyer in the sixth round, having a smaller gain corresponding to the cooperative class. The second seller becomes cooperative in the fourth negotiation round, while the first seller becomes cooperative in the fifth round. The first seller, classified as very cooperative, has the highest gain, with respect to the gain obtained in any other class. The second seller, classified as cooperative, has the highest gain, with respect to the gain obtained in any other class.

9.3.2. Four-to-Three Automated Negotiation Scenario

The following scenario implies a B2B automated negotiation with a pharmaceutical supplier, for two different medicines. During negotiation, the four buyers and the three sellers are changing dynamically their strategies.

In Figures 9.19 and 9.20 are represented the buyers and respectively the sellers gain versus the negotiation index.

From the Figures 9.19 and 9.20, it can be deduced that the gain increases a lot, when the agent is changing its negotiation strategy. When the agent is using the same strategy for many negotiation rounds, its gain remains almost the same or increases with small values.

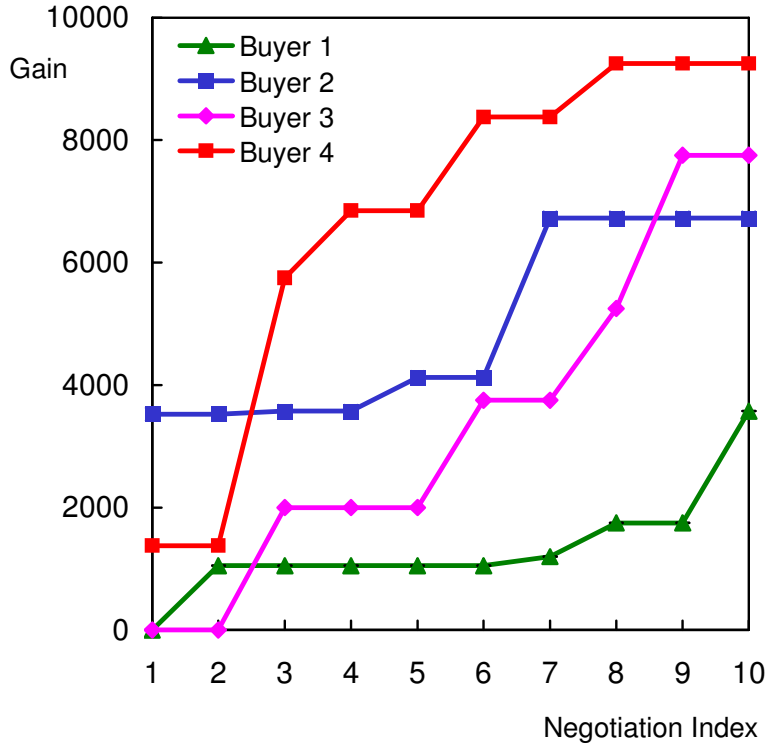


Figure 9.19. The Dependence of the Buyers Gain on the Negotiation Index

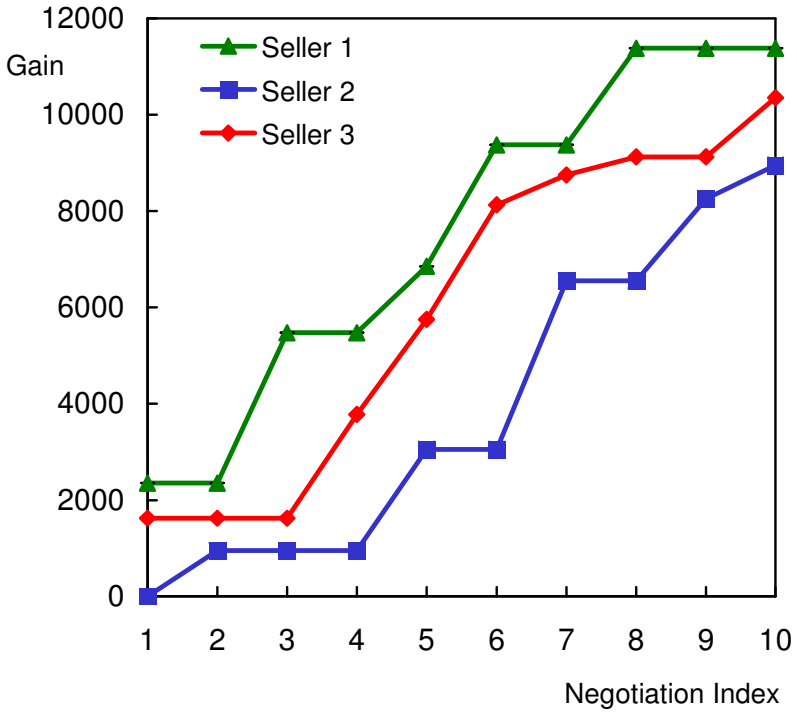


Figure 9.20. The Dependence of the Sellers Gain on the Negotiation Index

Chapter 10. Conclusions and Future Work

10.1. Conclusions

The current research on automated negotiation approaches some challenging problems concerning multi-attribute negotiation, bargaining using negotiation profiles, and negotiation involving cooperation classes for the agents. If there is incomplete information regarding the partner negotiating agent, it is often complex to compute agents' strategies. It is useful to design learning mechanisms for choosing certain strategies for agents to use.

Regarding multi-agents, it is necessary to understand the negotiation power, which is related to the relative abilities of agents in a situation to have influence over each other. In a bilateral negotiation, each agent's negotiation power is affected by its minimal and maximal prices, number of attributes bargained, negotiation deadline, etc. When many buyers and sellers are involved in negotiation, it is important to investigate how the market competition affects agents' negotiation strategies. With a great number of buyers and sellers, a single agent is unlikely to have much influence on the market equilibrium.

The automated negotiation has economical outcomes, because it has lower transaction costs, enabling higher volumes and new types of transactions in the electronic business domain. Through its automation, the negotiation mechanism becomes available to autonomous systems, improving the performance of these systems, when negotiation is used for agent coordination and cooperation, instead of existing interaction mechanisms.

In this thesis, a set of models for automated negotiation agents are developed, which are endowed with adaptive negotiation strategies. These models are implemented and the agent behavior is tested on different settings of B2B and B2C.

The first part of the thesis, Chapter 2 to Chapter 4, is dedicated to a review of the most important concepts, methods and techniques, which are relevant to the research approach and topics. In the same time, the challenges that lay ahead of the current research on agent negotiation are identified.

The second part of the thesis contains the personal contribution in research. In the model presented in Chapter 5, the agents' behavior is motivated by the gain they could get while satisfying their objectives and by the necessity to cooperate with other agents for obtaining these objectives. During cooperation and negotiation, the agent's beliefs on the other agents are updated, as the agent comes to know more about the others.

Because the agents' preferences are based on their needs, changes of their necessities influence the preferences during negotiation. The agents can modify their preferences over negotiation outcomes, when new information is available. The

communication primitives and the framework can express complex negotiation dialogues, in which agents change their preferences in time.

The agents use the Iterated Contract Net protocol, which has the advantage that it can simulate a real-world scenario, with many buyers and sellers having parallel negotiations. Because the negotiation complexity is an important issue at run-time, which can slow down the negotiation time, the agents' preferences are processed in the XML configuration file, before running the negotiations. The automated negotiation mechanism facilitates the self-interested agents to make decisions, which give them the optimal outcome.

An automated negotiation environment, which combines the agents' beliefs about the other agents in the system, with the possibility to represent and modify the negotiation strategy, is developed. The strategy is represented in the form of rules, with their attached preference coefficients. The negotiation strategy is improved in time using the Q-learning algorithm, applied upon the preference coefficients of the rules.

The tests performed show that the change of the values for the preference coefficients gives better results when using the reinforcement learning algorithm, than in the case when a predefined formula is used.

The rules expressed in Jess form a conflict set. There is a constraint solver, based on the preference coefficients of the rules, which solves the possible conflicts, and also has an important role in multi-attribute negotiations.

The agents' behavior can change during negotiation, according to previous interactions with other agents in the system. Changing behavior may refer to either the use of different negotiation strategies or to concessions made for other agents, with which they have successfully negotiated in the past. To this aim, an agent develops a set of profiles during negotiation: the preference profile, the partner cooperation profile, and the group-of-partners' negotiation profile. The first two profiles characterize individuals, while in a group negotiation profile, several agent profiles are clustered, according to commonly discovered features. Different approaches to the development of these profiles are presented in Chapter 5.

A set of negotiation strategies used by the agents is implemented in Chapter 6. They employ linear and non-linear negotiation strategies. The non-linear strategies can be divided into boulware strategy and conceder strategy. The experiments demonstrate different behavior and gain for the agents employing distinct strategies. The strategy used by agents is dependent on the number of buyers and sellers in the virtual market, and also on the business model, in which agents are acting.

The three possible negotiation strategies: linear, conceder, and boulware, can be dynamically changed, during run-time, for each negotiating agent, being either buyer or seller, using the graphical interface of each agent. The combination of strategies for buyer and seller agents gives different gains for the agents. For instance, when the buyer is using the conceder strategy and the seller employs the boulware strategy, then the buyer gain is increased.

The experimental results proved the expected results regarding the buyer agents. The higher gain is obtained for the buyer using the conceder strategy, the medium gain corresponds to the linear strategy and the smaller gain is obtained for the boulware strategy. Also, for the seller agents, the simulations enhanced the theoretical results. The higher gain is obtained for the seller using the boulware strategy, the medium gain corresponds to the linear strategy and the smaller gain is obtained for the conceder strategy.

Also, in Chapter 6 are described negotiation behaviors using weighted combinations of tactics for each one of the negotiation issues. The tests performed proved that it is better to use a mixture of resource-dependent tactics and behavior-dependent tactics.

A multi-agent system, based on the BDI model, which has rules for describing the agents' negotiation strategies, is described in Chapter 7. The rules that guide negotiation are based on price, and on the cooperation profile the agent develops during previous interactions with other agents in the system. The model allows the definition of several types of agents, by varying their behavior, according to the desire to obtain the maximum gain.

While negotiating with other agents, an agent develops cooperation profiles, which are gradually refined, and uses them to control its behavior. Better results are obtained after negotiation, if the agent's beliefs on other agents are updated periodically, as the agent knows more about the others. The introduction and gradually refinement of the cooperation profile of other agents represents a form of agent learning.

The use of the cooperation profiles is a challenging idea, and the cooperation classes, in which the agents are classified (unknown, non-cooperative, slightly cooperative, cooperative, very cooperative, highly cooperative), can be dynamically changed during the negotiation, with respect to the results obtained after each negotiation step.

Chapters 8 and 9 present the implementation of the multi-agent system for four business cases and reports different experimental results for each of this case. The system is tested for different business models, each model having its own particularities.

A travel agency business model and the associated negotiation rules used by the agents are described in Chapter 8. The gain obtained by agents during negotiation is computed after each negotiation step and is represented graphically for each negotiation. Also, the gain corresponding to each cooperation class of the agents is displayed. Different negotiations are performed, in which the number of buyers and sellers is gradually increased.

Chapter 9 describes some use cases involving different business models: a real estate agency scenario, a car dealer scenario, and an emergency hospital scenario. Different negotiation strategies are employed by agents: linear, conceder, and

boulware strategies. Different tests are performed and the results are graphically displayed.

Annexes 1, 2, and 3 describe in details the rules associated to the three business models presented in Chapter 9. Annex 1 presents the rules associated to the automated negotiation business model with a real estate agency. Annex 2 describes the rules from the business model involving a car dealer, while Annex 3 emphasizes the rules from an emergency hospital business model.

10.2. Contributions

The main original contributions of this thesis are the following:

- A model of self-interested agents acting in an open environment, which capture the most relevant elements of agents' behavior related to negotiation with other agents
- A framework for automated negotiation based on negotiation profiles and rules that encode the agents' negotiation strategy
- A set of negotiation profiles: the preference profile, the partner cooperation profile and the group-of-partners' negotiation profile, which characterize individuals and group of agents, clustered according to commonly discovered features
- Representation of the negotiation strategy in the form of production rules with associated preference coefficients to help select the most relevant negotiation rule
- Definition of the notion of cooperation classes for an agent and the classification of its cooperation potential, based on the C4.5 algorithm
- Two ways of updating the rule preference coefficients: (i) a heuristic formula obtained through experimental trials, and (ii) by means of a Q-learning algorithm, in which the learning performance is improved by state clusterization using k-means clustering algorithm
- A model of negotiating agents endowed with a set of negotiation strategies, from which the agent can learn to select the best one
- An associated multi-agent system in which, following the above model, an agent has the possibility to choose between three negotiation strategies: linear, conceder, and boulware, and also to evaluate the proposals using tactics
- Implementation of three possible negotiation strategies used by agents, in the form of a knowledge base with rules, written in Jess

- Evaluation of the proposed model in a settings consisting of agents that buy components from other agents, assemble these components and sell the assembled results to customers
- A multi-agent system, in which BDI agents are interacting towards reaching agreements for different negotiation objects, based on the utility of the negotiation objects
- Evaluation of the different possible roles the facilitator in the system may have, correlated with the cooperation profile of the agents
- Development of an open, scalable and adaptive multi-agent system for automated negotiation, implemented using Java, Jade, Jess, and XML, which supports different negotiation strategies and one-to-one, one-to-many, many-to-one, and many-to-many negotiations between self-interested agents
- Evaluation in the system, by different set of experiments, of the possibility to improve in time the negotiation strategy of the agents, as more negotiations are taking place, using machine learning techniques
- Evaluation of the proposed models in the system for four business cases:
    - automated negotiation in a travel agency scenario
    - automated negotiation between real estate agencies and the real estate developers
    - automated negotiation between car dealers negotiating with car factories
    - buyer agents representing emergency hospitals negotiating with medicines seller agents in a setting in which time is the most critical factor
- A state of the art account on negotiation in multi-agent systems, knowledge representation and learning, and also adaptive negotiation strategies

10.3. Publications

The following list contains the research papers developed during the PhD studies:

Book and Journal Papers

Serban Radu, Eugenia Kalisz, Adina Florea, "A Model of Automated Negotiation based on Agents Profiles", Scalable Computing: Practice and Experience Journal, 14, 1, 47-55, 2013 (BDI Journal)

Serban Radu, Eugenia Kalisz, Adina Florea, "Automatic Negotiation with Profiles and Clustering of Agents", International Journal of Intelligence Science, 3, 2, 69-76, 2013 (BDI Journal)

Serban Radu, “An Automated Negotiation System with Autonomous Agents for a Travel Agency Business Model”, Scientific Bulletin of the University Politehnica of Bucharest, Series C: Electrical Engineering and Computer Science, submitted, 2013 (BDI Journal)

Catalin Tudose, Carmen Odubasteanu, **Serban Radu**, "Java Reflection Performance Analysis Using Different Java Development", Advances in Intelligent Control Systems and Computer Science, 187, 439-452, 2013 (ISBN 978-3-642-32547-2)

Conference Proceedings Papers

Serban Radu, Valentin Lungu, “An Adaptive Multi-Agent Model for Automated Negotiation”, Proceedings of the 19-th International Conference on Control Systems and Computer Science, Bucharest, Romania, 29-31 May 2013, vol. 1, 167-174, 2013 (IEEE CPS)

Mihai Trascau, Teodor Tartareanu, Marius Benea, **Serban Radu**, “Emergence of Norms in Multi-Agent Societies; An Ultimatum Game Case Study”, Proceedings of the 5-th International Conference on Computational Collective Intelligence, Technologies and Applications, Craiova, Romania, 11-13 September 2013, To be published by Springer, (ISI Proceedings)

Luca Dan Serbanati, **Serban Radu**, "Paradigm Shifts in Health Informatics", Proceedings of 6-th International Conference on Health Informatics, Barcelona, Spain, 11-14 February 2013, 256-262, 2013 (ISI Proceedings)

Valentin Lungu, Andra Baltoiu, **Serban Radu**, “Using Emotion as Motivation in the Newtonian Emotion System”, Proceedings of the 7-th International Symposium on Intelligent Distributed Computing, Prague, Czech Republic, 4-6 September 2013, To be published by Springer, (ISI Proceedings)

Serban Radu, Eugenia Kalisz, Adina Florea, "Agents Negotiation Profiles for Automatic Transactions in Open Environments", The 14-th International Symposium on Symbolic and Numerical Algorithms for Scientific Computing, Timisoara, Romania, 26-29 September 2012

Adina Florea, **Serban Radu**, “Enhancing Pen-based Experiences with the Use of Concept Maps”, Proceedings of the 1-st International Workshop on Pen-Based Learning Technologies, Catania, Italy, 24-25 May 2007, IEEE Computer Society Conference Publishing Service CPS, 17-22, 2007 (ISI Proceedings)

Papers published in Proceedings of Summer Schools

Serban Radu, ”An Automated Negotiation Model based on Different Strategies in an Adaptive Multi-Agent System”, Proceedings of the 9-th International Summer School on Advanced Computer Architecture and Compilation for High-Performance and Embedded Systems, Fiuggi, Italy, 14-20 July 2013

Serban Radu, Adina Florea, "An Adaptive Multi-Agent System for e-Commerce", Proceedings of the 8-th International Summer School on Advanced Computer Architecture and Compilation for High-Performance and Embedded Systems, Fiuggi, Italy, 8-14 July 2012, 297-300, 2012

10.4. Future Work

A future research direction is to investigate an alternate approach to update the preference coefficients. This may be done using genetic algorithms. Moreover, genetic algorithms that use rule-specific genetic operators can be used to evolve new strategy rules, based on the existing ones.

Future research will include the design of new business models for creating virtual enterprises. A virtual enterprise refers to a temporary group of autonomous agents, which is formed to fulfill a certain objective or to give a special service. This will involve a series of negotiations among virtual enterprise agents.

Heuristic negotiation strategies used in this thesis are based on the exchange of proposals. The feedback that can be received from the opponent is a counterproposal. The argumentation-based negotiation extends the negotiation protocols with the possibility to exchange arguments. This information gives explicitly the opinion of the agent making the argument. Future work will investigate the arguments an agent should use, in order to improve the negotiation outcomes.

A future research direction is to discover new applications of automated negotiation. Information incompleteness and the existence of market competition make it difficult to compute agents’ equilibrium strategies. An agent needs to learn from its negotiation history. Market dynamics may require an agent to reason about future trading opportunities. In addition, each agent needs to reason about other agents’ strategies.

In addition to design negotiation strategies that maximize an agent’s utility, creating negotiation mechanisms that maximize some global performance measures, like social welfare, is also a future research direction. One line of research refers at investigating some simplified bargaining games. The other line of research aims at considering more complex environments and evaluating different mechanisms through experimentation.

Another interesting future research direction is bargaining in trading networks. Different from trading in markets, a buyer and a seller can negotiate for an

agreement if and only if they have a relationship, to engage in exchange. This setting is practical, because individual buyers and sellers could trade through intermediaries and not all buyers and sellers could interact with the same intermediaries.

Another related future research direction is building systems to support human negotiation, which is difficult due to a number of reasons. First, it is necessary to consider much larger negotiation space and strategy space. For instance, human beings often use body language while doing negotiation. Second, it is necessary to consider many other factors, such as emotion, trust, power, and culture.

Also, in the future it is expected to identify possible partnerships with researchers from the academic and business environments, in order to exploit the results obtained by the multi-agent system platform for automated negotiation.

Bibliography

1. An B., Nicola G., Lesser V., Extending Alternating-Offers Bargaining in One-to-Many and Many-to-Many Settings, Proceedings of the 9-th IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT), 423-453, 2009
2. An B., Lesser V., Sim K.M., Strategic Agents for Multi-Resource Negotiation, Auton Agent Multi-Agent Systems, Springer, DOI 10.1007/s10458-010-9137-2, 2010
3. Arthur D., Vassilvitskii S., K-Means++: The Advantages of Careful Seeding, Proceedings of the 8-th Annual ACM-SIAM Symposium on Discrete Algorithms SODA, 1027-1035, 2007
4. Badica C., Badita A., Ganzha M., Iordache A., Paprzycki M., Rule-Based Framework for Automated Negotiation: Initial Implementation, Proceedings of the RuleML, 193-198, 2005
5. Badica C., Badita A., Ganzha M., Implementing Rule-based Mechanisms for Agent-based Price Negotiations, Proceedings of the 2006 ACM Symposium on Applied Computing, 96-100, 2006 a
6. Badica C., Ganzha M., Paprzycki M., Rule-based Automated Price Negotiation: an Overview and an Experiment, Proceedings of International Conference on Artificial Intelligence and Soft Computing, 1050-1059, 2006 b
7. Badica C., Ganzda M., Paprzycki M., Implementing Rule-based Automated Price Negotiation in an Agent System, Journal of Universal Computer Science, 13, 2, 244-266, 2007
8. Bartolini C., Preist C., Jennings N.R., A Generic Software Framework for Automated Negotiation, Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems AAMAS, 213-235, 2005
9. Beam C., Segev A., Automated Negotiations: A Survey of the State of the Art, Wirtschaftsinformatik 39, 3, 263-267, 1997
10. Bellifemine F., Caire G., Greenwood D., Developing Multi-Agent Systems with JADE, John Wiley and Sons Ltd., 2007
11. Benameur H., Chaib-draa B., Kropf P., Multi-item Auctions for Automatic Negotiation, Journal of Information and Software Technologies, Elsevier, 44/5, 291-301, 2002
12. Bratman M.E., Intention, Plans, and Practical Reason, CSLI Publications, 1999
13. Brzostowski J., Kowalczyk R., Adaptive Negotiation with On-line Prediction of Opponent Behaviour in Agent-based Negotiations, Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology IAT'06, 2006
14. Bull L., Learning Classifier Systems: A Brief Introduction, Studies in Fuzziness and Soft Computing, 150, 1-12, 2004
15. Butz M.V., Learning Classifier Systems, GECCO 2010, Proceedings of the 12-th Annual Conference Companion on Genetic and Evolutionary Computation, 2331-2352, 2010
16. Carabelea C., Adaptive Agents in Argumentation-Based Negotiation, Springer Verlag, LNAI series, 2002
17. Chavez A., Maes P., An Agent Marketplace for Buying and Selling Goods, Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, 1996
18. Chen Y., Peng Y., Finin T., Labrou Y., Cost S., A Negotiation-based Multi-agent System for Supply Chain Management, Proceedings of Agents 99 Workshop on Agent Based Decision-Support for Managing the Internet-Enabled Supply-Chain, 1999
19. Choi H.R., Kim H.S., Hong S.G., Park Y.J., Park Y.S., Kang M.H., Implementation of Framework for Developing Multi-Agent based Automated Negotiation Systems, Proceedings of the 7-th International Conference on Electronic Commerce, ICEC'05, 306-315, 2005
20. Coehoorn R., Jennings N., Learning an Opponent's Preferences to Make Effective Multi-Issue Negotiation Trade-Offs, Proceedings of the 6-th International Conference on Electronic Commerce, 59-68, 2004

21. Crawford E., Veloso M., Learning to Select Negotiation Strategies in Multi-Agent Meeting Scheduling, Working notes of the Multiagent Learning Workshop, Proceedings of the AAAI, 584-595, 2005
22. Dong H., Hussain F.K., Chang E., State of the Art in Negotiation Ontologies for Multi-agent Systems, International Journal of Web Services Practices, 3, 3-4, 157-163, 2008
23. Faratin P., Sierra C., Jennings N.R., Negotiation Decision Functions for Autonomous Agents, International Journal of Robotics and Autonomous Systems, 24, 3-4, 159-182, 1998
24. Fatima S.S., Wooldridge M., Jennings N.R., Multi-issue Negotiation with Deadlines, Journal of Artificial Intelligence Research 27, 381-417, 2006
25. Fatima S.S., Wooldridge M., Jennings N.R., Approximate and Online Multi-Issue Negotiation, Proceedings of the 6-th International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS'07, 951-958, 2007
26. Fatima S.S., Wooldridge M., Jennings N.R., On Optimal Agendas for Multi-Issue Negotiation, Proceedings of the 12-th International Workshop on Agent-Mediated Electronic Commerce, AMEC 2010, 155-168, 2010
27. Filzmoser M., Simulation of Automated Negotiation, Springer-Verlag Vienna, 2010
28. Florea A.M., Using Utility Values in Argument-based Negotiation, Proceedings of IC-AI'02, International Conference on Artificial Intelligence, CSREA Press, 1021-1026, 2002
29. Florea A.M., Kalisz E., Adaptive Negotiation Based on Rewards and Regret in a Multi-agent Environment, Proceedings of the 9-th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, IEEE Computer Society Press, 254-259, 2007
30. Florea A.M., Radu S., Enhancing Pen-based Experiences with the Use of Concept Maps, Proceedings of 1-st International Workshop on Pen-Based Learning Technologies, IEEE Computer Society Conference Publishing Service CPS, 17-22, 2007
31. Florea A.M., Radu S., Mogos A., Prolog Programming Techniques for Artificial Intelligence, Printech Publishing House, 2007
32. Florea A.M., Kalisz E., A Negotiation Learning Model for Open Multi-Agent Environments, International Journal of Computing Anticipatory Systems IJCAS 20, CHAOS, 20, 121-130, 2008
33. Florea A.M., Mocanu I., Mogos A., Urzica A., Radu S., SCIPA Research Report, 23-38, 2008
34. Florea A.M., Multi-Agent Systems, Lecture Notes, 2012
35. Friedman-Hill E., Jess in Action: Rule-Based Systems in Java, Manning Publications Co., 2003
36. Georgeff M., Pell B., Pollack M., Tambe M., Wooldridge M., The Belief-Desire-Intention Model of Agency, Intelligent Agents V: Agents Theories, Architectures and Languages, Lecture Notes in Computer Science, 1555, 1-10, 1999
37. Goradia H.J, Vidal J.M., An Equal Excess Negotiation Algorithm for Coalition Formation, Proceedings of the 6-th International Joint Conference on Autonomous Agents and Multiagents Systems, 1059-1061, 2007
38. Groves W., Collins J., Gini M., Visualization and Analysis Methods for Comparing Agent Behavior in TAC-SCM, Proceedings of the 8-th International Conference on Autonomous Agents and Multiagent Systems, 2, 1367-1368, 2009
39. Guttman R.H., Moukes A.G., Maes P., Agent-Mediated Electronic Commerce: A Survey, The Knowledge Engineering Review 13, 2, 147-159, 1998
40. Guttman R.H., Maes P., Cooperative vs. Competitive Multi-Agent Negotiations in Retail Electronic Commerce, Proceedings of the 2-nd International Workshop on Cooperative Information Agents, CIA'98, 1998
41. He M., Jennings N.R., Leung H.-F., On Agent-Mediated Electronic Commerce, IEEE Transactions on Knowledge and Data Engineering 15, 4, 985-1003, 2003

42. Hindriks K., Jonker C.M., Tykhonov D., The Benefits of Opponent Models in Negotiation, Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, 2, 439-444, 2009
43. Horrocks I., Patel-Schneider P.F., van Harmelen F., From SHIQ and RDF to OWL: the Making of a Web Ontology Language, Journal of Web Semantics, 1, 1, 7-26, 2003
44. Huq G., Automated Negotiation in Multi-Agent based Electronic Business, VDM Verlag, 2010
45. Ito T., Hattori H., Klein M., Multi-issue Negotiation Protocol for Agents: Exploring Nonlinear Utility Spaces, Proceedings of the 20-th International Joint Conference on Artificial Intelligence, IJCAI 2007, 1347-1352, 2007
46. Jennings N.R., Norman T.J., Faratin P., O'Brien P., Odgers B., Autonomous Agents for Business Process Management, Applied Artificial Intelligence, 14, 2, 145-189, 2000
47. Jennings N.R., Faratin P., Lomuscio A.R., Parsons S., Sierra C., Wooldridge M., Automated Negotiation: Prospects, Methods and Challenges, International Journal of Group Decision and Negotiation, 10, 199-215, 2001
48. Jonker C., Robu V., Treur J., An Agent Architecture for Multi-attribute Negotiation using Incomplete Preference Information, Autonomous Agents and Multi-Agent Systems, 15, 221-252, 2007
49. Kaelbling L.P., Littman M.L., Moore A.W., Reinforcement Learning: A Survey, Journal of Artificial Intelligence Research, 4, 237-285, 1996
50. Kakas A., Moraitis P., Adaptive Agent Negotiation via Argumentation, Proceeding of the AAMAS'06 Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, 384-391, 2006
51. Kholief M., Nada N., Khedr W., Ontology-Oriented Inference-Based Learning Content Management System, International Journal of Web and Semantic Technology, 3, 3, 131-142, 2012
52. Kowalczyk R., Bui V., On Constraint-based Reasoning in e-Negotiation Agents, Agent Mediated Electronic Commerce III, LNAI, Dijmem F., Cortés U. eds., Springer-Verlag, 31-46, 2000
53. Kraus S., Sycara K., Evenchik A., Reaching Agreements through Argumentation: a Logical Model and Implementation, Artificial Intelligence, Elsevier Science, 104, 1-69, 1998
54. Kraus S., Automated Negotiation and Decision Making in Multiagent Environments, Proceedings of the ACAI 2001, LNAI 2086, 150-172, 2001
55. Kumar S., Agent-Based Semantic Web Service Composition, SpringerBriefs in Electrical and Computer Engineering, 2012
56. Lau R.Y.K., Adaptive Negotiation Agents for E-business, Proceedings of the 7-th International Conference on Electronic Commerce ICEC'05, 271-278, 2005
57. Lau R.Y.K., Tang M., Wong O., Stephen W. Milliner S.W., Chen Y-P.P., An Evolutionary Learning Approach for Adaptive Negotiation Agents, International Journal of Intelligent Systems, 21, 41-72, 2006
58. Lin R., Kraus S., Wilkenfeld J., Barry J., Negotiation with Bounded Rational Agents in Environments with Incomplete Information using an Automated Agent, Journal of Artificial Intelligence, 172, 6-7, 823-851, 2008
59. Lin R., Kraus S., Tykhonov D., Hindriks K., Jonker C.M., Supporting the Design of General Automated Negotiators, Proceedings of the 2-nd International Workshop on Agent-based Complex Automated Negotiations, ACAN'09, 1-20, 2009 a
60. Lin R., Oshrat Y., Kraus S., Investigating the Benefits of Automated Negotiations in Enhancing People's Negotiation Skills, Proceedings of the 8-th International Conference on Automated Agents and Multi-agent Systems, 1, 345-352, 2009 b
61. Lin R., Kraus S., Can Automated Agents Proficiently Negotiate with Humans?, Communications of the ACM, 53, 1, 78-88, 2010

62. Lungu V., Baltoiu A., Radu S., Using Emotion as Motivation in the Newtonian Emotion System, Proceedings of the 7-th International Symposium on Intelligent Distributed Computing, 2013
63. Manju S., Punithavalli M., An Analysis of Q-Learning Algorithms with Strategies of Reward Functions, International Journal on Computer Science and Engineering, 3, 2, 814-820, 2011
64. Mazid M.M., Ali A.S. and Ticke K.S., Improved C4.5 Algorithm for Rule Based Classification, Proceedings of the 9-th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases, AIKED'10, 296-301, 2010
65. Mercier H., Sperber D., Why do Humans Reason? Arguments for an Argumentative Theory, Behavioral and Brain Sciences, 34, 57-111, 2011
66. Mocanu I., Kalisz E., Negreanu L., Genetic Algorithms Viewed as Anticipatory Systems, Proceedings of the 9-th International Conference on Computing Anticipatory Systems, 1303, 207-215, 2010
67. Murthy S.K., Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey, Data Mining and Knowledge Discovery, 2, 345-389, 1998
68. Narayanan V., Jennings N.R., An Adaptive Bilateral Negotiation Model for E-Commerce Settings, Proceedings of the 7-th International IEEE Conference on E-Commerce Technology, 34-39, 2005
69. Naroditskiy V., Polukarov M., Jennings N.R., Optimal Payments in Dominant-strategy Mechanisms for Single-parameter Domains, ACM Transactions on Economics and Computation, 1, 4, 1-21, 2013
70. Noy N.F., McGuinness D.L., Ontology Development 101: A Guide to Creating Your First Ontology, Stanford Knowledge Systems Laboratory Technical Report, KSL-01-05, 2001
71. Osborne M.J., Rubinstein A., A Course in Game Theory, MIT Press, 1994
72. Osborne M.J., An Introduction to Game Theory, New York, Oxford: Oxford University Press, 2004
73. Pena J.M., Lozano J.A., Larranaga P., An Empirical Comparison of Four Initialization Methods for the K-Means Algorithm, Journal Pattern Recognition Letters, 20, 10, 1027-1040, 1999
74. Quinlan J.R., C 4. 5: Programs for Machine Learning, Morgan Kaufmann Publishers, USA, 1993
75. Radu S., Florea A.M., An Adaptive Multi-Agent System for e-Commerce, Proceedings of Advanced Computer Architecture and Compilation for High-Performance and Embedded Systems, ACACES 2012, Academia Press, 297-300, 2012
76. Radu S., Kalisz E., Florea A.M., A Model of Automated Negotiation Based on Agents Profiles, Scalable Computing: Practice and Experience Journal, 14, 1, 47-55, 2013 a
77. Radu S., Kalisz E., Florea A.M., Automatic Negotiation with Profiles and Clustering of Agents, International Journal of Intelligence Science, 3, 2, 2013 b
78. Radu S., Lungu V., An Adaptive Multi-Agent Model for Automated Negotiation, Proceedings of the 19-th International Conference on Control Systems and Computer Science, 1, 167-174, 2013
79. Radu S., An Automated Negotiation Model based on Different Strategies in an Adaptive Multi-Agent System, Proceedings of the 9-th International Summer School on Advanced Computer Architecture and Compilation for High-Performance and Embedded Systems, 2013 a
80. Radu S., An Automated Negotiation System with Autonomous Agents for a Travel Agency Business Model, Scientific Bulletin of the University Politehnica of Bucharest, Series C: Electrical Engineering and Computer Science, submitted, 2013 b
81. Rahwan I., Kowalczyk R., Pham H.H., Intelligent Agents for Automated One-to-Many e-Commerce Negotiation, Proceedings of the 25-th Australasian Conference on Computer Science, 197-203, 2002 a

82. Rahwan I., Ramchurn S.D., Jennings N.R., McBurney P., Parsons S., Sonenberg L., Argumentation-Based Negotiation, Proceedings of the ACSC – 25-th Australasian Conference on Computer Science, 197-203, 2002 b
83. Rahwan I., Ramchurn S.D., Jennings N.R., McBurney P., Parsons S., Sonenberg L., Argumentation-Based Negotiation, The Knowledge Engineering Review, 18, 343-375, 2003
84. Rosenschein J.S., Zlotkin G., Rules of Encounter, MIT Press, Cambridge, USA, 1994
85. Sandholm T., Lesser V., Leveled Commitment Contracts and Strategic Breach, Games and Economic Behavior, 35, 1–2, 212–270, 2001
86. Serbanati L.D., Radu S., Paradigm Shifts in Health Informatics, Proceedings of the 6-th International Conference on Health Informatics, 256-262, 2013
87. Shoham Y., Leyton-Brown K., Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations, Cambridge University Press, 2009
88. Sierra C., Faratin P., Jennings N.R., A Service-Oriented Negotiation Model between Autonomous Agents, Lecture Notes in Computer Science, 1237, 17-35, 1997
89. Silva A., Neto J.A., Ibert I., A Computation Environment for Automated Negotiation: A Case Study in Electronic Tourism, Proceedings of the 22-nd ACM Symposium on Applied Computing, SAC'07, 654-658, 2007
90. Silva S., Collective Bargaining Negotiation, International Labour Organisation, ACT/EMP Publications, 1-16, 1996
91. Skylogiannis T., Antoniou G., Bassiliades N., Governatori G., Bikakis A., Dr-Negotiate: A System for Automated Agent Negotiation with Defeasible Logic-Based Strategies, Journal Data & Knowledge Engineering, 63, 2, 362-380, 2007
92. Smith R.G., The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver, Journal IEEE Transactions on Computer, 29, 11, 1104-1113, 1980
93. Soh L.-K., Li X., Adaptive, Confidence-Based Multiagent Negotiation Strategy, Proceedings of the 3-rd International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS'04, 1048-1055, 2004
94. Srivihok A., Sukonmanee P., E-commerce Intelligent Agent: Personalization Travel Support Agent Using Q Learning", Proceedings of the 7-th International Conference on Electronic Commerce, ICEC 2005, 287-292, 2005
95. Tamma V., Phelpsa S., Dickinson I., Wooldridge M., Ontologies for Supporting Negotiation in e-Commerce, Engineering Applications of Artificial Intelligence, 18, 223–236, 2005
96. Tesauro G., Kephart J., Pricing in Agent Economies Using Multi-Agent Q-Learning, Journal Autonomous Agents and Multi-Agent Systems, 5, 3, 289-304, 2002
97. Trascau M., Tartareanu T.A., Benea M.T., Radu S., Establishing Social Order Through Norm Emergence, Proceedings of the 5-th International Conference on Computational Collective Intelligence, Technologies and Applications, Craiova, Romania, 2013
98. Tu M.T., Wolff E., Lamersdorf W., Genetic Algorithms for Automated Negotiations: A FSMbased Application Approach, Proceedings of the 11-th International Workshop on Database and Expert Systems Applications DEXA00, 1029-1033, 2000
99. Tudose C., Odubasteanu C., Radu S., Java Reflection Performance Analysis Using Different Java Development, Advances in Intelligent Control Systems and Computer Science, 187, 439-452, 2013
100. Wellman M. P., Wurman, P. R., Market-aware agents for a multiagent world, Robotics and Autonomous Systems, 24, 115–125, 1998
101. Wooldridge M., An Introduction to MultiAgent Systems, John Wiley & Sons, Ltd, Publication 2002
102. Wooldridge M., An Introduction to MultiAgent Systems, Second Edition, John Wiley & Sons, Ltd, Publication 2009

103. Wunder M., Kaisers M., Yaros J. R., Littman M., Using Iterated Reasoning to Predict Opponent Strategies, Proceedings of the 10-th International Conference on Autonomous Agents and Multiagent Systems, 2, 593-600, 2011
104. Ye Y., Liu J., Moukas A., Agents in Electronic Commerce, Electronic Commerce Research 1,1-2, 9-14, 2001
105. Zeng D., Sycara K., Bayesian Learning in Negotiation, International Journal of Human-Computer Studies, 48, 1, 125-141, 1998
106. Zhang S., Makedon F., Privacy Preserving Learning in Negotiation, Proceedings of the 2005 ACM Symposium on Applied Computing, 821-825, 2005
107. Zlatev Z., Diakov N., Pokraev S., Construction of Negotiation Protocols for e-Commerce Applications, ACM SIGecom Exchanges, 5, 2, 12-22, 2004

Annex 1. Real Estate Agency Automated Negotiation Rules

The **Jess** rules upon which the negotiation is performed in the real estate agency business model are described in what follows. First the rules of the buyer agent are presented, and then the rules associated to the seller agent.

The rules assciated to the **ACCEPT** communication primitive of the buyer are first defined.

1) The first **ACCEPT** rule says that an offer for a quantity *q* for a certain product having the price between *minPrice* and *maxPrice*, from an unknown or non-cooperative partner is accepted, if the price is less than *q*(minPrice+maxPrice)/2*;

```
(defrule accept1
  (NegotiationObject {sellerClassification == "nc" ||
sellerClassification == "u"}(currentOffer ?o) (quantity ?q))
  (Product (minPrice ?minPrice) (maxPrice ?maxPrice &: (<= ?o (* ?q (/
(+ ?minPrice ?maxPrice) 2)))))
    => (add (new Offer "accept1" ?*accept*)))
```

2) The second **ACCEPT** rule deals with the case when an offer for a quantity *q* for a certain product having the price between *minPrice* and *maxPrice*, from a slightly cooperative or cooperative partner is accepted, if the price is between *q*(minPrice+maxPrice)/2* and *q*maxPrice*;

```
(defrule accept2
  (NegotiationObject {sellerClassification == "sc" ||
sellerClassification == "c"}(currentOffer ?o) (quantity ?q))
  (Product (minPrice ?minPrice) (maxPrice ?maxPrice &: (<= ?o (* ?q
?maxPrice)) &: (> ?o (* ?q (/ (+ ?minPrice ?maxPrice) 2)))))
    => (add (new Offer "accept2" ?*accept*)))
```

3) The third **ACCEPT** rule deals with the case when an offer for a quantity *q* for a certain product having the price between *minPrice* and *maxPrice*, from a very cooperative or highly cooperative partner is accepted, if the price is less or equal than *q*maxPrice*;

```
(defrule accept3
  (NegotiationObject {sellerClassification == "vc" ||
sellerClassification == "hc"}(currentOffer ?o) (quantity ?q))
  (Product (minPrice ?minPrice) (maxPrice ?maxPrice &: (<= ?o (* ?q
?maxPrice))))
    => (add (new Offer "accept3" ?*accept*)))
```

For the **REJECT** communication primitive of the buyer, there is the following rule:

1) The buyer will reject an offer for a negotiation object, received after a certain time threshold, which is defined as *5* seconds for each negotiation. The message *Negotiation time elapsed* is sent from the buyer to the seller;

```
(defrule reject1
  (NegotiationObject (timeElapsed ?te &: (>= ?te ?*maxNegTime*)))
    => (add (new Offer "reject1" ?*reject* "Negotiation time
elapsed.")))
```

For the **PROPOSE** communication primitive of the buyer, there are the following rules:

1) The price offered by a buyer, in the beginning, for a certain quantity *q* of products, having the price between *minPrice* and *maxPrice*, to a non-cooperative or unknown seller, is equal to *q*minPrice*;

```
(defrule firstPrice1
  (NegotiationObject {step == 1} {sellerClassification == "nc" ||
sellerClassification == "u"} (quantity ?q))
  (Product (minPrice ?minPrice))
    => (add (new Offer "firstPrice1" (* ?q ?minPrice) ?*propose*)))
```

2) The price offered by a buyer, in the beginning, for a certain quantity *q* of products, having the price between *minPrice* and *maxPrice*, to a slightly cooperative or cooperative seller, is equal to *q*(minPrice+500)*;

```
(defrule firstPrice2
  (NegotiationObject {step == 1} {sellerClassification == "sc" ||
sellerClassification == "c"} (quantity ?q))(Product (minPrice ?minPrice))
    => (add (new Offer "firstPrice2" (* ?q (+ 500 ?minPrice))
?*propose*)))
```

3) The price offered by a buyer, in the beginning, for a certain quantity *q* of products, having the price between *minPrice* and *maxPrice*, to a very cooperative or highly cooperative seller, is equal to *q*(minPrice+1000)*;

```
(defrule firstPrice3
  (NegotiationObject {step == 1} {sellerClassification == "vc" ||
sellerClassification == "hc"} (quantity ?q))(Product (minPrice ?minPrice))
    => (add (new Offer "firstPrice3" (* ?q (+ 1000 ?minPrice))
?*propose*)))
```

For the **linear strategy** of the buyer, there are the following rules:

1) The buyer will increase its previous offer *o* with *200*, to the non-cooperative or unknown sellers, when the negotiation step is greater than *1*;

```
(defrule newPrice4
  (declare (salience 10))
  (NegotiationObject {step > 1} (previousOffer ?o)
{sellerClassification == "nc" || sellerClassification == "u"}
        (quantity ?q)) (Strategy {strategy == ?*linear*})
  (Product (maxPrice ?maxPrice &: (> (* ?q ?maxPrice) (+ ?o (* 200
?q))))) => (add (new Offer "newPrice4" (+ ?o (* 200 ?q)) ?*propose*)))
```

2) The buyer will increase its previous offer *o* with *400*, to the slightly cooperative or cooperative sellers, when the negotiation step is greater than *1*;

```
(defrule newPrice5
  (NegotiationObject {step > 1} (previousOffer ?o)
{sellerClassification == "sc" || sellerClassification == "c"}
        (quantity ?q)) (Strategy {strategy == ?*linear*})
  (Product (maxPrice ?maxPrice &: (> (* ?q ?maxPrice) (+ ?o (* 400
?q))))) => (add (new Offer "newPrice5" (+ ?o (* 400 ?q)) ?*propose*)))
```

3) The buyer will increase its previous offer *o* with *600*, to the very cooperative or highly cooperative sellers, when the negotiation step is greater than *1*;

```
(defrule newPrice6
  (NegotiationObject {step > 1} (previousOffer ?o)
{sellerClassification == "vc" || sellerClassification == "hc"}
        (quantity ?q)) (Strategy {strategy == ?*linear*})
  (Product (maxPrice ?maxPrice &: (> (* ?q ?maxPrice) (+ ?o (* 600
?q))))) => (add (new Offer "newPrice6" (+ ?o (* 600 ?q)) ?*propose*)))
```

For the **conceder strategy** of the buyer, there are the following rules:

1) The buyer will increase its previous offer *o* with *1500*, to the non-cooperative or unknown sellers, when the negotiation step is greater than *1* and lower than *12*;

```
(defrule newPrice7
  (NegotiationObject {step > 1 && step < ?*concederRounds*}
(previousOffer ?o) {sellerClassification == "nc" || sellerClassification ==
"u"}(quantity ?q)) (Strategy {strategy == ?*conceder*})
  (Product (maxPrice ?maxPrice &: (> (* ?q ?maxPrice) (+ ?o (* 1500
?q))))) => (add (new Offer "newPrice7" (+ ?o (* 1500 ?q)) ?*propose*)))
```

2) The buyer will increase its previous offer *o* with *2000*, to the slightly cooperative or cooperative buyers, when the negotiation step is greater than *1* and lower than *12*;

```
(defrule newPrice8
 (NegotiationObject {step > 1 && step < ?*concederRounds*}
(previousOffer ?o) {sellerClassification == "sc" || sellerClassification ==
"c"}(quantity ?q)) (Strategy {strategy == ?*conceder*})
 (Product (maxPrice ?maxPrice &: (> (* ?q ?maxPrice) (+ ?o (* 2000
?q))))) => (add (new Offer "newPrice8" (+ ?o (* 2000 ?q)) ?*propose*)))
```

3) The buyer will increase its previous offer *o* with *2500*, to the very cooperative or highly cooperative sellers, when the negotiation step is greater than *1* and lower than *12*;

```
(defrule newPrice9
 (NegotiationObject {step > 1 && step < ?*concederRounds*}
(previousOffer ?o) {sellerClassification == "vc" || sellerClassification ==
"hc"}(quantity ?q)) (Strategy {strategy == ?*conceder*})
 (Product (maxPrice ?maxPrice &: (> (* ?q ?maxPrice) (+ ?o (* 2500
?q))))) => (add (new Offer "newPrice9" (+ ?o (* 2500 ?q)) ?*propose*)))
```

4) The buyer will increase its previous offer *o* with *400*, to the non-cooperative or unknown sellers, when the negotiation step is greater than *12*;

```
(defrule newPrice10
 (NegotiationObject {step >= ?*concederRounds*} (previousOffer ?o)
{sellerClassification == "nc" || sellerClassification == "u"}
        (quantity ?q)) (Strategy {strategy == ?*conceder*})
 (Product (maxPrice ?maxPrice &: (> (* ?q ?maxPrice) (+ ?o (* 400
?q))))) => (add (new Offer "newPrice10" (+ ?o (* 400 ?q)) ?*propose*)))
```

5) The buyer will increase its previous offer *o* with *600*, to the slightly cooperative or cooperative sellers, when the negotiation step is greater than *12*;

```
(defrule newPrice11
 (NegotiationObject {step >= ?*concederRounds*} (previousOffer ?o)
{sellerClassification == "sc" || sellerClassification == "c"}
        (quantity ?q)) (Strategy {strategy == ?*conceder*})
 (Product (maxPrice ?maxPrice &: (> (* ?q ?maxPrice) (+ ?o (* 600
?q))))) => (add (new Offer "newPrice11" (+ ?o (* 600 ?q)) ?*propose*)))
```

6) The buyer will increase its previous offer *o* with *800*, to the very cooperative or highly cooperative sellers, when the negotiation step is greater than *12*;

```
(defrule newPrice12
 (NegotiationObject {step >= ?*concederRounds*} (previousOffer ?o)
{sellerClassification == "vc" || sellerClassification == "hc"}
        (quantity ?q)) (Strategy {strategy == ?*conceder*})
 (Product (maxPrice ?maxPrice &: (> (* ?q ?maxPrice) (+ ?o (* 800
?q))))) => (add (new Offer "newPrice12" (+ ?o (* 8 ?q)) ?*propose*)))
```

For the **boulware strategy** of the buyer, there are the following rules:

1) The buyer will increase its previous offer *o* with *1*, when the negotiation step is greater than *1* and the time elapsed in the negotiation is less than *4* seconds, represented by the global variable *boulwareTime1*;

```
(defrule newPrice13
 (NegotiationObject {step > 1} (previousOffer ?o) {timeElapsed <
?*boulwareTime1*}(quantity ?q)) (Strategy {strategy == ?*boulware*})
    => (add (new Offer "newPrice13" (+ ?o (* 1 ?q)) ?*propose*)))
```

2) The buyer will increase its previous offer *o* with *1000*, when the negotiation step is greater than *1* and the time elapsed in the negotiation is greater than *4* seconds, represented by the global variable *boulwareTime1*;

```
(defrule newPrice14
 (NegotiationObject {step > 1} (previousOffer ?o) {timeElapsed >=
?*boulwareTime1*}(quantity ?q)) (Strategy {strategy == ?*boulware*})
 (Product (maxPrice ?maxPrice &: (> (* ?q ?maxPrice) (+ ?o (* 1000
?q))))) => (add (new Offer "newPrice14" (+ ?o (* 1000 ?q)) ?*propose*)))
```

3) The buyer will increase its previous offer *o* with *3000*, when the negotiation step is greater than *1* and the time elapsed in the negotiation is greater than *4.5* seconds, represented by the global variable *boulwareTime2*;

```
(defrule newPrice15
 (NegotiationObject {step > 1} (previousOffer ?o) {timeElapsed >=
?*boulwareTime2*}(quantity ?q)) (Strategy {strategy == ?*boulware*})
 (Product (maxPrice ?maxPrice &: (> (* ?q ?maxPrice) (+ ?o (* 3000
?q))))) => (add (new Offer "newPrice15" (+ ?o (* 3000 ?q)) ?*propose*)))
```

For the seller, regarding the **ACCEPT** communication primitive, there are the following rules:

1) The seller will accept an offer from a non-cooperative or unknown buyer, for a certain quantity of *q* products, having the price between *minPrice* and *maxPrice*, with the price greater than *q*(minPrice+maxPrice)/2*;

```
(defrule accept1
 (NegotiationObject {buyerClassification == "nc" ||
buyerClassification == "u"} {step > 0}(currentOffer ?o) (quantity ?q))
 (Product (minPrice ?minPrice) (maxPrice ?maxPrice &: (>= ?o (* ?q (/
(+ ?minPrice ?maxPrice) 2))))) => (add (new Offer "accept1" ?*accept*)))
```

2) The seller will accept an offer from a slightly cooperative or cooperative buyer, for a certain quantity of *q* products, having the price between *minPrice* and *maxPrice*, with the price less than *q*(maxPrice-(minPrice+maxPrice)/2)*;

```
(defrule accept2
 (NegotiationObject {buyerClassification == "sc" ||
buyerClassification == "c"} {step > 0}(currentOffer ?o) (quantity ?q))
 (Product (minPrice ?minPrice) (maxPrice ?maxPrice &: (<= ?o (* ?q
?maxPrice)) &: (> ?o (* ?q (/ (+ ?minPrice ?maxPrice) 2)))))
   => (add (new Offer "accept2" ?*accept*)))
```

3) The seller will accept an offer from a very cooperative or highly cooperative buyer, for a certain quantity of *q* products, having the price between *minPrice* and *maxPrice*, with the price lower than *q*maxPrice*;

```
(defrule accept3
 (NegotiationObject {buyerClassification == "vc" ||
buyerClassification == "hc"} {step > 0}(currentOffer ?o) (quantity ?q))
 (Product (minPrice ?minPrice) (maxPrice ?maxPrice &: (<= ?o (* ?q
?maxPrice)))) => (add (new Offer "accept3" ?*accept*)))
```

For the **REJECT** communication primitive of the seller, there are the following rules:

1) The seller will reject an offer for a negotiation object, received after a certain time threshold, which is defined as *5* seconds for each negotiation. The message *Negotiation time elapsed* is sent from the seller to the buyer;

```
(defrule reject1
 (NegotiationObject (timeElapsed ?te &: (>= ?te ?*maxNegTime*)))
   => (add (new Offer "reject1" ?*reject* "Negotiation time
elapsed.")))
```

For the **PROPOSE** communication primitive of the seller, there are the following rules:

1) The price proposed by a seller, in the beginning, for a certain quantity *q* of products, having the price between *minPrice* and *maxPrice*, to a non-cooperative or unknown buyer, is equal to *q*maxPrice*;

```
(defrule firstPrice1
  (NegotiationObject {step == 0} {buyerClassification == "nc" ||
buyerClassification == "u"} (quantity ?q))(Product (maxPrice ?maxPrice))
    => (add (new Offer "firstPrice1" (* ?maxPrice ?q) ?*propose*)))
```

2) The price proposed by a seller, in the beginning, for a certain quantity *q* of products, having the price between *minPrice* and *maxPrice*, to a slightly cooperative or cooperative buyer, is equal to *q*(maxPrice-500)*;

```
(defrule firstPrice2
  (NegotiationObject {step == 0} {buyerClassification == "sc" ||
buyerClassification == "c"} (quantity ?q))(Product (maxPrice ?maxPrice))
    => (add (new Offer "firstPrice2" (* (- ?maxPrice 500) ?q)
?*propose*)))
```

3) The price proposed by a seller, in the beginning, for a certain quantity *q* of products, having the price between *minPrice* and *maxPrice*, to a very cooperative or highly cooperative buyer, is equal to *q*(maxPrice-1000)*;

```
(defrule firstPrice3
  (NegotiationObject {step == 0} {buyerClassification == "vc" ||
buyerClassification == "hc"} (quantity ?q))(Product (maxPrice ?maxPrice)  )
    => (add (new Offer "firstPrice3" (* (- ?maxPrice 1000) ?q)
?*propose*)))
```

For the **linear strategy** of the seller, there are the following rules:

1) The seller will decrease its previous offer *o* with *200*, to the non-cooperative or unknown buyers, when the negotiation step is greater than *1*;

```
(defrule newPrice4
  (NegotiationObject {step > 1} (previousOffer ?o) {buyerClassification
== "nc" || buyerClassification == "u"}(quantity ?q))
  (Product (minPrice ?minPrice &: (<= (* ?q ?minPrice) (- ?o (* 200
?q))))) (Strategy {strategy == ?*linear*})
    => (add (new Offer "newPrice4" (- ?o (* 200 ?q)) ?*propose*)))
```

2) The seller will decrease its previous offer *o* with *400*, to the slightly cooperative or cooperative buyers, when the negotiation step is greater than *1*;

```
(defrule newPrice5
  (NegotiationObject {step > 1} (previousOffer ?o) {buyerClassification
== "sc" || buyerClassification == "c"}
      (quantity ?q)) (Strategy {strategy == ?*linear*})
  (Product (minPrice ?minPrice &: (<= (* ?q ?minPrice) (- ?o (* 400
?q))))) =>  (add (new Offer "newPrice5" (- ?o (* 400 ?q)) ?*propose*)))
```

3) The seller will decrease its previous offer *o* with *600*, to the very cooperative or highly cooperative buyers, when the negotiation step is greater than *1*;

```
(defrule newPrice6
  (NegotiationObject {step > 1} (previousOffer ?o) {buyerClassification
== "vc" || buyerClassification == "hc"}
      (quantity ?q)) (Strategy {strategy == ?*linear*})
  (Product (minPrice ?minPrice &: (<= (* ?q ?minPrice) (- ?o (* 600
?q))))) =>  (add (new Offer "newPrice6" (- ?o (* 600 ?q)) ?*propose*)))
```

For the **conceder strategy** of the seller, there are the following rules:

1) The seller will decrease its previous offer *o* with *1500*, to the non-cooperative or unknown buyers, when the negotiation step is greater than *1* and lower than *12*;

```
(defrule newPrice7
```

```
      (NegotiationObject {step > 1 && step < ?*concederRounds*}
(previousOffer ?o) {buyerClassification == "nc" || buyerClassification ==
"u"}(quantity ?q))
      (Product (minPrice ?minPrice &: (<= (* ?q ?minPrice) (- ?o (* 1500
?q))))) (Strategy {strategy == ?*conceder*})
        => (add (new Offer "newPrice7" (- ?o (* 1500 ?q)) ?*propose*)))
```

2) The seller will decrease its previous offer *o* with *2000*, to the slightly cooperative or cooperative buyers, when the negotiation step is greater than *1* and lower than *12*;

```
     (defrule newPrice8
      (NegotiationObject {step > 1 && step < ?*concederRounds*}
(previousOffer ?o) {buyerClassification == "sc" || buyerClassification ==
"c"}(quantity ?q))
      (Product (minPrice ?minPrice &: (<= (* ?q ?minPrice) (- ?o (* 2000
?q))))) (Strategy {strategy == ?*conceder*})
        => (add (new Offer "newPrice8" (- ?o (* 2000 ?q)) ?*propose*)))
```

3) The seller will decrease its previous offer *o* with *2500*, to the very cooperative or highly cooperative buyers, when the negotiation step is greater than *1* and lower than *12*;

```
     (defrule newPrice9
      (NegotiationObject {step > 1 && step < ?*concederRounds*}
(previousOffer ?o) {buyerClassification == "vc" || buyerClassification ==
"hc"}(quantity ?q)) (Product (minPrice ?minPrice))
      (Strategy {strategy == ?*conceder*})
      (test (<= (* ?q ?minPrice) (- ?o (* 2500 ?q))))
        => (add (new Offer "newPrice9" (- ?o (* 2500 ?q)) ?*propose*)))
```

4) The seller will decrease its previous offer *o* with *q*400*, to the non-cooperative or unknown buyers, when the negotiation step is greater than *12;*

```
     (defrule newPrice10
      (NegotiationObject {step >= ?*concederRounds*} (previousOffer ?o)
{buyerClassification == "nc" || buyerClassification == "u"}(quantity ?q))
      (Product (minPrice ?minPrice &: (<= (* ?q ?minPrice) (- ?o (* 400
?q))))) (Strategy {strategy == ?*conceder*})
        => (add (new Offer "newPrice10" (- ?o (* 400 ?q)) ?*propose*)))
```

5) The seller will decrease its previous offer *o* with *q*600*, to the slightly cooperative or cooperative buyers, when the negotiation step is greater than *12*;

```
     (defrule newPrice11
      (NegotiationObject {step >= ?*concederRounds*} (previousOffer ?o)
{buyerClassification == "sc" || buyerClassification == "c"}(quantity ?q))
      (Product (minPrice ?minPrice &: (<= (* ?q ?minPrice) (- ?o (* 600
?q))))) (Strategy {strategy == ?*conceder*})
        => (add (new Offer "newPrice11" (- ?o (* 600 ?q)) ?*propose*)))
```

6) The seller will decrease its previous offer *o* with *q*800*, to the very cooperative or highly cooperative buyers, when the negotiation step is greater than *12*;

```
     (defrule newPrice12
      (NegotiationObject {step >= ?*concederRounds*} (previousOffer ?o)
{buyerClassification == "vc" || buyerClassification == "hc"}
            (quantity ?q)) (Strategy {strategy == ?*conceder*})
      (Product (minPrice ?minPrice &: (<= (* ?q ?minPrice) (- ?o (* 800
?q))))) =>  (add (new Offer "newPrice12" (- ?o (* 800 ?q)) ?*propose*)))
```

For the **boulware strategy** of the seller, there are the following rules:

1) The seller will decrease its previous offer *o* by *1*, when the negotiation step is greater than *1* and the time elapsed in the negotiation is less than *4* seconds, represented by the global variable *boulwareTime1*;

```
(defrule newPrice13
  (NegotiationObject {step > 1} (previousOffer ?o) {timeElapsed < ?*boulwareTime1*}(quantity ?q)) (Strategy {strategy == ?*boulware*})
    => (add (new Offer "newPrice13" (- ?o ?q) ?*propose*)))
```

2) The seller will decrease its previous offer *o* by *1000*, when the negotiation step is greater than *1* and the time elapsed in the negotiation is greater than *4* seconds, represented by the global variable *boulwareTime1*;

```
(defrule newPrice14
  (NegotiationObject {step > 1} (previousOffer ?o) {timeElapsed >= ?*boulwareTime1*} (quantity ?q))
  (Product (minPrice ?minPrice &: (<= (* ?q ?minPrice) (- ?o (* 1000 ?q))))) (Strategy {strategy == ?*boulware*})
    => (add (new Offer "newPrice14" (- ?o (* 1000 ?q)) ?*propose*)))
```

3) The seller will decrease its previous offer *o* by *2000*, when the negotiation step is greater than *1* and the time elapsed in the negotiation is greater than *4.5* seconds, represented by the global variable *boulwareTime2*;

```
(defrule newPrice15
  (NegotiationObject {step > 1} (previousOffer ?o) {timeElapsed >= ?*boulwareTime2*}
        (quantity ?q)) (Strategy {strategy == ?*boulware*})
  (Product (minPrice ?minPrice &: (<= (* ?q ?minPrice) (- ?o (* 2000 ?q))))) =>  (add (new Offer "newPrice15" (- ?o (* 2000 ?q)) ?*propose*)))
```

Annex 2. Car Dealer Automated Negotiation Rules

The **Jess** rules upon which the negotiation is performed in the car dealer business model are described in this annex. First the rules of the buyer agent are presented, and then the rules associated to the seller agent.

The rules associated to the **ACCEPT** communication primitive of the buyer are first defined.

1) The first **ACCEPT** rule says that an offer for a quantity *q* for a certain product having the price between *minPrice* and *maxPrice*, from an unknown or non-cooperative partner is accepted, if the price is less than *q*(minPrice+maxPrice)/2*;

```
(defrule accept1
  (NegotiationObject {sellerClassification == "nc" || sellerClassification == "u"}(currentOffer ?o) (quantity ?q))
  (Product (minPrice ?minPrice) (maxPrice ?maxPrice &: (<= ?o (* ?q (/ (+ ?minPrice ?maxPrice) 2))))) =>  (add (new Offer "accept1" ?*accept*)))
```

2) The second **ACCEPT** rule deals with the case when an offer for a quantity *q* for a certain product having the price between *minPrice* and *maxPrice*, from a slightly cooperative or cooperative partner is accepted, if the price is between *q*(minPrice+maxPrice)/2* and *q*maxPrice*;

```
(defrule accept2
  (NegotiationObject {sellerClassification == "sc" || sellerClassification == "c"}(currentOffer ?o) (quantity ?q))
  (Product (minPrice ?minPrice) (maxPrice ?maxPrice &: (<= ?o (* ?q ?maxPrice)) &: (> ?o (* ?q (/ (+ ?minPrice ?maxPrice) 2)))))
    => (add (new Offer "accept2" ?*accept*)))
```

3) The third **ACCEPT** rule deals with the case when an offer for a quantity *q* for a certain product having the price between *minPrice* and *maxPrice*, from a very cooperative or highly cooperative partner is accepted, if the price is less or equal than *q*maxPrice*;

```
(defrule accept3
  (NegotiationObject {sellerClassification == "vc" || sellerClassification == "hc"}(currentOffer ?o) (quantity ?q))
  (Product (minPrice ?minPrice) (maxPrice ?maxPrice &: (<= ?o (* ?q ?maxPrice)))) =>  (add (new Offer "accept3" ?*accept*)))
```

For the **REJECT** communication primitive of the buyer, there is the following rule:

1) The buyer will reject an offer for a negotiation object, received after a certain time threshold, which is defined as *5* seconds for each negotiation. The message *Negotiation time elapsed* is sent from the buyer to the seller;

```
(defrule reject1
  (NegotiationObject (timeElapsed ?te &: (>= ?te ?*maxNegTime*)))
   =>  (add  (new  Offer  "reject1"  ?*reject*  "Negotiation  time elapsed.")))
```

For the **PROPOSE** communication primitive of the buyer, there are the following rules:

1) The price offered by a buyer, in the beginning, for a certain quantity *q* of products, having the price between *minPrice* and *maxPrice*, to a non-cooperative or unknown seller, is equal to *q*minPrice*;

```
(defrule firstPrice1
  (NegotiationObject {step == 1} {sellerClassification == "nc" || sellerClassification == "u"} (quantity ?q))(Product (minPrice ?minPrice))
    => (add (new Offer "firstPrice1" (* ?q ?minPrice) ?*propose*)))
```

2) The price offered by a buyer, in the beginning, for a certain quantity *q* of products, having the price between *minPrice* and *maxPrice*, to a slightly cooperative or cooperative seller, is equal to *q*(minPrice+50)*;

```
(defrule firstPrice2
```

```
      (NegotiationObject {step == 1} {sellerClassification == "sc" ||
sellerClassification == "c"} (quantity ?q))(Product (minPrice ?minPrice))
        => (add (new Offer "firstPrice2" (* ?q (+ 50 ?minPrice))
?*propose*)))
```

3) The price offered by a buyer, in the beginning, for a certain quantity *q* of products, having the price between *minPrice* and *maxPrice*, to a very cooperative or highly cooperative seller, is equal to *q*(minPrice+100)*;

```
     (defrule firstPrice3
      (NegotiationObject {step == 1} {sellerClassification == "vc" ||
sellerClassification == "hc"} (quantity ?q))(Product (minPrice ?minPrice))
        => (add (new Offer "firstPrice3" (* ?q (+ 100 ?minPrice))
?*propose*)))
```

For the **linear strategy** of the buyer, there are the following rules:

1) The buyer will increase its previous offer *o* with *100*, to the non-cooperative or unknown sellers, when the negotiation step is greater than *1*;

```
     (defrule newPrice4
      (NegotiationObject     {step     >     1}     (previousOffer     ?o)
{sellerClassification == "nc" || sellerClassification == "u"}
            (quantity ?q)) (Strategy {strategy == ?*linear*})
      (Product (maxPrice ?maxPrice))
        => (if (> (* ?q ?maxPrice) (+ ?o (* 100 ?q))) then
            (add (new Offer "newPrice4" (+ ?o (* 100 ?q)) ?*propose*))
         else (add (new Offer "newPrice4" (* ?q ?maxPrice) ?*propose*))))
```

2) The buyer will increase its previous offer *o* with *150*, to the slightly cooperative or cooperative sellers, when the negotiation step is greater than *1*;

```
     (defrule newPrice5
      (NegotiationObject     {step     >     1}     (previousOffer     ?o)
{sellerClassification == "sc" || sellerClassification == "c"}
            (quantity ?q)) (Strategy {strategy == ?*linear*})
      (Product (maxPrice ?maxPrice))
        => (if (> (* ?q ?maxPrice) (+ ?o (* 150 ?q))) then
            (add (new Offer "newPrice5" (+ ?o (* 150 ?q)) ?*propose*))
         else (add (new Offer "newPrice5" (* ?q ?maxPrice) ?*propose*))))
```

3) The buyer will increase its previous offer *o* with *200*, to the very cooperative or highly cooperative sellers, when the negotiation step is greater than *1*;

```
     (defrule newPrice6
      (NegotiationObject     {step     >     1}     (previousOffer     ?o)
{sellerClassification == "vc" || sellerClassification == "hc"}
            (quantity ?q)) (Strategy {strategy == ?*linear*})
      (Product (maxPrice ?maxPrice))
        => (if (> (* ?q ?maxPrice) (+ ?o (* 200 ?q))) then
            (add (new Offer "newPrice6" (+ ?o (* 200 ?q)) ?*propose*))
         else (add (new Offer "newPrice6" (* ?q ?maxPrice) ?*propose*))))
```

For the **conceder strategy** of the buyer, there are the following rules:

1) The buyer will increase its previous offer *o* with *700*, to the non-cooperative or unknown sellers, when the negotiation step is greater than *1* and lower than *12*;

```
     (defrule newPrice7
      (NegotiationObject  {step  >  1  &&  step  <  ?*concederRounds*}
(previousOffer ?o) {buyerClassification == "nc" || buyerClassification ==
"u"}(quantity ?q)) (Product (maxPrice ?maxPrice))
      (Strategy {strategy == ?*conceder*})
```

```
        => (if (> (* ?q ?maxPrice) (+ ?o (* 700 ?q))) then
            (add (new Offer "newPrice7" (+ ?o (* 700 ?q)) ?*propose*))
         else (add (new Offer "newPrice7" (* ?q ?maxPrice) ?*propose*))))
```

2) The buyer will increase its previous offer *o* with *800*, to the slightly cooperative or cooperative buyers, when the negotiation step is greater than *1* and lower than *12*;

```
    (defrule newPrice8
     (NegotiationObject {step > 1 && step < ?*concederRounds*}
(previousOffer ?o) {buyerClassification == "sc" || buyerClassification ==
"c"}(quantity ?q)) (Product (maxPrice ?maxPrice))
     (Strategy {strategy == ?*conceder*})
       => (if (> (* ?q ?maxPrice) (+ ?o (* 800 ?q))) then
           (add (new Offer "newPrice8" (+ ?o (* 800 ?q)) ?*propose*))
        else (add (new Offer "newPrice8" (* ?q ?maxPrice) ?*propose*))))
```

3) The buyer will increase its previous offer *o* with *900*, to the very cooperative or highly cooperative sellers, when the negotiation step is greater than *1* and lower than *12*;

```
    (defrule newPrice9
     (NegotiationObject {step > 1 && step < ?*concederRounds*}
(previousOffer ?o) {buyerClassification == "vc" || buyerClassification ==
"hc"}(quantity ?q)) (Product (maxPrice ?maxPrice))
     (Strategy {strategy == ?*conceder*})
       => (if (> (* ?q ?maxPrice) (+ ?o (* 900 ?q))) then
           (add (new Offer "newPrice9" (+ ?o (* 900 ?q)) ?*propose*))
        else (add (new Offer "newPrice9" (* ?q ?maxPrice) ?*propose*))))
```

4) The buyer will increase its previous offer *o* with *100*, to the non-cooperative or unknown sellers, when the negotiation step is greater than *12*;

```
    (defrule newPrice10
     (NegotiationObject {step >= ?*concederRounds*} (previousOffer ?o)
{sellerClassification == "nc" || sellerClassification == "u"}
           (quantity ?q)) (Strategy {strategy == ?*conceder*})
     (Product (maxPrice ?maxPrice))
       => (if (> (* ?q ?maxPrice) (+ ?o (* 100 ?q))) then
           (add (new Offer "newPrice10" (+ ?o (* 100 ?q)) ?*propose*))
        else (add (new Offer "newPrice10" (* ?q ?maxPrice) ?*propose*))))
```

5) The buyer will increase its previous offer *o* with *150*, to the slightly cooperative or cooperative sellers, when the negotiation step is greater than *12*;

```
    (defrule newPrice11
     (NegotiationObject {step >= ?*concederRounds*} (previousOffer ?o)
{sellerClassification == "sc" || sellerClassification == "c"}
           (quantity ?q)) (Strategy {strategy == ?*conceder*})
     (Product (maxPrice ?maxPrice))
       => (if (> (* ?q ?maxPrice) (+ ?o (* 150 ?q))) then
           (add (new Offer "newPrice11" (+ ?o (* 150 ?q)) ?*propose*))
        else (add (new Offer "newPrice11" (* ?q ?maxPrice) ?*propose*))))
```

6) The buyer will increase its previous offer *o* with *200*, to the very cooperative or highly cooperative sellers, when the negotiation step is greater than *12*;

```
    (defrule newPrice12
     (NegotiationObject {step >= ?*concederRounds*} (previousOffer ?o)
{sellerClassification == "vc" || sellerClassification == "hc"}
           (quantity ?q)) (Strategy {strategy == ?*conceder*})
     (Product (maxPrice ?maxPrice))
       => (if (> (* ?q ?maxPrice) (+ ?o (* 200 ?q))) then
```

```
        (add (new Offer "newPrice12" (+ ?o (* 200 ?q)) ?*propose*))
      else (add (new Offer "newPrice12" (* ?q ?maxPrice) ?*propose*))))
```

For the **boulware strategy** of the buyer, there are the following rules:

1) The buyer will increase its previous offer *o* with *1*, when the negotiation step is greater than *1* and the time elapsed in the negotiation is less than *4* seconds, represented by the global variable *boulwareTime1*;

```
(defrule newPrice13
  (NegotiationObject {step > 1} (previousOffer ?o) {timeElapsed <
?*boulwareTime1*}(quantity ?q)) (Strategy {strategy == ?*boulware*})
    => (add (new Offer "newPrice13" (+ ?o (* 1 ?q)) ?*propose*)))
```

2) The buyer will increase its previous offer *o* with *800*, when the negotiation step is greater than *1* and the time elapsed in the negotiation is greater than *4* seconds, represented by the global variable *boulwareTime1*;

```
(defrule newPrice14
  (NegotiationObject {step > 1} (previousOffer ?o) {timeElapsed >=
?*boulwareTime1*}(quantity ?q)) (Strategy {strategy == ?*boulware*})
  (Product (maxPrice ?maxPrice))
    => (if (> (* ?q ?maxPrice) (+ ?o (* 800 ?q))) then
        (add (new Offer "newPrice14" (+ ?o (* 800 ?q)) ?*propose*))
      else (add (new Offer "newPrice14" (* ?q ?maxPrice) ?*propose*))))
```

3) The buyer will increase its previous offer *o* with *1000*, when the negotiation step is greater than *1* and the time elapsed in the negotiation is greater than *4.5* seconds, represented by the global variable *boulwareTime2*;

```
(defrule newPrice15
  (NegotiationObject {step > 1} (previousOffer ?o) {timeElapsed >=
?*boulwareTime2*}(quantity ?q)) (Strategy {strategy == ?*boulware*})
  (Product (maxPrice ?maxPrice))
    => (if (> (* ?q ?maxPrice) (+ ?o (* 1000 ?q))) then
        (add (new Offer "newPrice14" (+ ?o (* 1000 ?q)) ?*propose*))
      else (add (new Offer "newPrice14" (* ?q ?maxPrice) ?*propose*))))
```

For the seller, regarding the **ACCEPT** communication primitive, there are the following rules:

1) The seller will accept an offer from a non-cooperative or unknown buyer, for a certain quantity of *q* products, having the price between *minPrice* and *maxPrice*, with the price greater than *q*(minPrice+maxPrice)/2*;

```
(defrule accept1
  (NegotiationObject {buyerClassification == "nc" ||
buyerClassification == "u"} {step > 0}(currentOffer ?o) (quantity ?q))
  (Product (minPrice ?minPrice) (maxPrice ?maxPrice &: (>= ?o (* ?q (/
(+ ?minPrice ?maxPrice) 2))))) =>  (add (new Offer "accept1" ?*accept*)))
```

2) The seller will accept an offer from a slightly cooperative or cooperative buyer, for a certain quantity of *q* products, having the price between *minPrice* and *maxPrice*, with the price between *q*(minPrice+maxPrice)/2* and *q*maxPrice*;

```
(defrule accept2
  (NegotiationObject {buyerClassification == "sc" ||
buyerClassification == "c"} {step > 0}(currentOffer ?o) (quantity ?q))
  (Product (minPrice ?minPrice) (maxPrice ?maxPrice &: (<= ?o (* ?q
?maxPrice)) &: (> ?o (* ?q (/ (+ ?minPrice ?maxPrice) 2)))))
    => (add (new Offer "accept2" ?*accept*)))
```

3) The seller will accept an offer from a very cooperative or highly cooperative buyer, for a certain quantity of *q* products, having the price between *minPrice* and *maxPrice*, with the price lower than or equal to *q*maxPrice*;

```
(defrule accept3
  (NegotiationObject {buyerClassification == "vc" || buyerClassification == "hc"} {step > 0}(currentOffer ?o) (quantity ?q))
  (Product (minPrice ?minPrice) (maxPrice ?maxPrice &: (<= ?o (* ?q ?maxPrice)))) => (add (new Offer "accept3" ?*accept*)))
```

For the **REJECT** communication primitive of the seller, there are the following rules:

1) The seller will reject an offer for a negotiation object, received after a certain time threshold, which is defined as *5* seconds for each negotiation. The message *Negotiation time elapsed* is sent from the seller to the buyer;

```
(defrule reject1
  (NegotiationObject (timeElapsed ?te &: (>= ?te ?*maxNegTime*)))
    => (add (new Offer "reject1" ?*reject* "Negotiation time elapsed.")))
```

For the **PROPOSE** communication primitive of the seller, there are the following rules:

1) The price proposed by a seller, in the beginning, for a certain quantity *q* of products, having the price between *minPrice* and *maxPrice*, to a non-cooperative or unknown buyer, is equal to:

a) *q*maxPrice*, if *q≤10*;

b) *q*maxPrice*0.95*, if *10<q<20*;

c) *q*maxPrice*0.85*, if *q≥20*.

```
(defrule firstPrice1
  (NegotiationObject {step == 0} {buyerClassification == "nc" || buyerClassification == "u"} (quantity ?q))(Product (maxPrice ?maxPrice))
    => (if (and (> ?q 10) (< ?q 20)) then
        (add (new Offer "firstPrice1" (* (* ?maxPrice 0.95) ?q) ?*propose*)) elif (>= ?q 20) then
        (add (new Offer "firstPrice1" (* (* ?maxPrice 0.85) ?q) ?*propose*)) else
        (add (new Offer "firstPrice1" (* ?maxPrice ?q) ?*propose*))))
```

2) The price proposed by a seller, in the beginning, for a certain quantity *q* of products, having the price between *minPrice* and *maxPrice*, to a slightly cooperative or cooperative buyer, is equal to:

a) *q*(maxPrice-150)*, if *q≤10*;

b) *q*maxPrice*0.9*, if *10<q<20*;

c) *q*maxPrice*0.8*, if *q≥20*.

```
(defrule firstPrice2
  (NegotiationObject {step == 0} {buyerClassification == "sc" || buyerClassification == "c"} (quantity ?q))(Product (maxPrice ?maxPrice) )
    => (if (and (> ?q 10) (< ?q 20)) then
        (add (new Offer "firstPrice2" (* (* ?maxPrice 0.90) ?q) ?*propose*)) elif (> ?q 20) then
        (add (new Offer "firstPrice2" (* (* ?maxPrice 0.80) ?q) ?*propose*)) else
        (add (new Offer "firstPrice2" (* (- ?maxPrice 150) ?q) ?*propose*))))
```

3) The price proposed by a seller, in the beginning, for a certain quantity *q* of products, having the price between *minPrice* and *maxPrice*, to a very cooperative or highly cooperative buyer, is equal to:

a) *q*(maxPrice-250)*, if *q≤10*;

b) *q*maxPrice*0.85*, if *10<q<20*;

c) *q*maxPrice*0.75*, if *q≥20*.

```
(defrule firstPrice3
```

```
      (NegotiationObject {step == 0} {buyerClassification == "vc" ||
buyerClassification == "hc"} (quantity ?q))(Product (maxPrice ?maxPrice))
        => (if (and (> ?q 10) (< ?q 20)) then
            (add (new Offer "firstPrice3" (* (* ?maxPrice 0.85) ?q)
?*propose*)) elif (> ?q 20) then
            (add (new Offer "firstPrice3" (* (* ?maxPrice 0.75) ?q)
?*propose*)) else
            (add (new Offer "firstPrice3" (* (- ?maxPrice 250) ?q)
?*propose*))))
```

For the **linear strategy** of the seller, there are the following rules:

1) The seller will decrease its previous offer *o* with *100*, to the non-cooperative or unknown buyers, when the negotiation step is greater than *1*;

```
     (defrule newPrice4
      (NegotiationObject {step > 1} (previousOffer ?o) {buyerClassification
== "nc" || buyerClassification == "u"} (quantity ?q)) (Product (minPrice
?minPrice)) (Strategy {strategy == ?*linear*})
        => (if (<= (* ?q ?minPrice) (- ?o (* 100 ?q))) then
            (add (new Offer "newPrice4" (- ?o (* 100 ?q)) ?*propose*))
         else (add (new Offer "newPrice4" (* ?q ?minPrice) ?*propose*))))
```

2) The seller will decrease its previous offer *o* with *150*, to the slightly cooperative or cooperative buyers, when the negotiation step is greater than *1*;

```
     (defrule newPrice5
      (NegotiationObject {step > 1} (previousOffer ?o) {buyerClassification
== "sc" || buyerClassification == "c"} (quantity ?q)) (Product (minPrice
?minPrice)) (Strategy {strategy == ?*linear*})
        => (if (<= (* ?q ?minPrice) (- ?o (* 150 ?q))) then
            (add (new Offer "newPrice5" (- ?o (* 150 ?q)) ?*propose*))
         else (add (new Offer "newPrice5" (* ?q ?minPrice) ?*propose*))))
```

3) The seller will decrease its previous offer *o* with *200*, to the very cooperative or highly cooperative buyers, when the negotiation step is greater than *1*;

```
     (defrule newPrice6
      (NegotiationObject {step > 1} (previousOffer ?o) {buyerClassification
== "vc" || buyerClassification == "hc"} (quantity ?q)) (Product (minPrice
?minPrice)) (Strategy {strategy == ?*linear*})
        => (if (<= (* ?q ?minPrice) (- ?o (* 200 ?q))) then
            (add (new Offer "newPrice6" (- ?o (* 200 ?q)) ?*propose*))
         else (add (new Offer "newPrice6" (* ?q ?minPrice) ?*propose*))))
```

For the **conceder strategy** of the seller, there are the following rules:

1) The seller will decrease its previous offer *o* with *700*, to the non-cooperative or unknown buyers, when the negotiation step is greater than *1* and lower than *12*;

```
     (defrule newPrice7
      (NegotiationObject {step > 1 && step < ?*concederRounds*}
(previousOffer ?o) {buyerClassification == "nc" || buyerClassification ==
"u"}(quantity ?q)) (Product (minPrice ?minPrice))
     (Strategy {strategy == ?*conceder*})
        => (if (<= (* ?q ?minPrice) (- ?o (* 700 ?q))) then
            (add (new Offer "newPrice7" (- ?o (* 700 ?q)) ?*propose*))
         else (add (new Offer "newPrice7" (* ?q ?minPrice) ?*propose*))))
```

2) The seller will decrease its previous offer *o* with *800*, to the slightly cooperative or cooperative buyers, when the negotiation step is greater than *1* and lower than *12*;

```
     (defrule newPrice8
```

```
      (NegotiationObject {step > 1 && step < ?*concederRounds*}
(previousOffer ?o) {buyerClassification == "sc" || buyerClassification ==
"c"}(quantity ?q)) (Product (minPrice ?minPrice))
      (Strategy {strategy == ?*conceder*})
        => (if (<= (* ?q ?minPrice) (- ?o (* 800 ?q))) then
            (add (new Offer "newPrice8" (- ?o (* 800 ?q)) ?*propose*))
         else (add (new Offer "newPrice8" (* ?q ?minPrice) ?*propose*))))
```

3) The seller will decrease its previous offer *o* with *900*, to the very cooperative or highly cooperative buyers, when the negotiation step is greater than *1* and lower than *12*;

```
    (defrule newPrice9
      (NegotiationObject {step > 1 && step < ?*concederRounds*}
(previousOffer ?o) {buyerClassification == "vc" || buyerClassification ==
"hc"}(quantity ?q)) (Product (minPrice ?minPrice))
      (Strategy {strategy == ?*conceder*})
        => (if (<= (* ?q ?minPrice) (- ?o (* 900 ?q))) then
            (add (new Offer "newPrice9" (- ?o (* 900 ?q)) ?*propose*))
         else (add (new Offer "newPrice9" (* ?q ?minPrice) ?*propose*))))
```

4) The seller will decrease its previous offer *o* with *100*, to the non-cooperative or unknown buyers, when the negotiation step is greater than *12*;

```
    (defrule newPrice10
      (NegotiationObject {step >= ?*concederRounds*} (previousOffer ?o)
{buyerClassification == "nc" || buyerClassification == "u"}
            (quantity ?q)) (Product (minPrice ?minPrice))
      (Strategy {strategy == ?*conceder*})
        => (if (<= (* ?q ?minPrice) (- ?o (* 100 ?q))) then
            (add (new Offer "newPrice10" (- ?o (* 100 ?q)) ?*propose*))
         else (add (new Offer "newPrice10" (* ?q ?minPrice) ?*propose*))))
```

5) The seller will decrease its previous offer *o* with *150*, to the slightly cooperative or cooperative buyers, when the negotiation step is greater than *12*;

```
    (defrule newPrice11
      (NegotiationObject {step >= ?*concederRounds*} (previousOffer ?o)
{buyerClassification == "sc" || buyerClassification == "c"}
            (quantity ?q)) (Product (minPrice ?minPrice))
      (Strategy {strategy == ?*conceder*})
        => (if (<= (* ?q ?minPrice) (- ?o (* 150 ?q))) then
            (add (new Offer "newPrice11" (- ?o (* 150 ?q)) ?*propose*))
         else (add (new Offer "newPrice11" (* ?q ?minPrice) ?*propose*))))
```

6) The seller will decrease its previous offer *o* with *200*, to the very cooperative or highly cooperative buyers, when the negotiation step is greater than *12*;

```
    (defrule newPrice12
      (NegotiationObject {step >= ?*concederRounds*} (previousOffer ?o)
{buyerClassification == "vc" || buyerClassification == "hc"}
            (quantity ?q)) (Product (minPrice ?minPrice))
      (Strategy {strategy == ?*conceder*})
        => (if (<= (* ?q ?minPrice) (- ?o (* 200 ?q))) then
            (add (new Offer "newPrice12" (- ?o (* 200 ?q)) ?*propose*))
         else (add (new Offer "newPrice12" (* ?q ?minPrice) ?*propose*))))
```

For the **boulware strategy** of the seller, there are the following rules:

1) The seller will decrease its previous offer *o* by *1*q*, when the negotiation step is greater than *1* and the time elapsed in the negotiation is less than *4* seconds, represented by the global variable *boulwareTime1*;

```
(defrule newPrice13
  (NegotiationObject {step > 1} (previousOffer ?o) {timeElapsed <
?*boulwareTime1*}(quantity ?q)) (Strategy {strategy == ?*boulware*})
    => (add (new Offer "newPrice13" (- ?o ?q) ?*propose*)))
```

2) The seller will decrease its previous offer *o* by *800*q*, when the negotiation step is greater than *1* and the time elapsed in the negotiation is greater than *4* seconds, represented by the global variable *boulwareTime1*;

```
(defrule newPrice14
  (NegotiationObject {step > 1} (previousOffer ?o) {timeElapsed >=
?*boulwareTime1*}(quantity ?q)) (Product (minPrice ?minPrice))
  (Strategy {strategy == ?*boulware*})
    => (if (<= (* ?q ?minPrice) (- ?o (* 800 ?q))) then
        (add (new Offer "newPrice14" (- ?o (* 800 ?q)) ?*propose*))
     else (add (new Offer "newPrice14" (* ?q ?minPrice) ?*propose*))))
```

3) The seller will decrease its previous offer *o* by *1000*q*, when the negotiation step is greater than *1* and the time elapsed in the negotiation is greater than *4.5* seconds, represented by the global variable *boulwareTime2*;

```
(defrule newPrice15
  (NegotiationObject {step > 1} (previousOffer ?o) {timeElapsed >=
?*boulwareTime2*}(quantity ?q)) (Strategy {strategy == ?*boulware*})
  (Product (minPrice ?minPrice))
    => (if (<= (* ?q ?minPrice) (- ?o (* 1000 ?q))) then
        (add (new Offer "newPrice15" (- ?o (* 1000 ?q)) ?*propose*))
     else (add (new Offer "newPrice15" (* ?q ?minPrice) ?*propose*))))
```

Annex 3. Emergency Hospital Automated Negotiation Rules

The **Jess** rules upon which the negotiation is performed in the emergency hospital business model are described in this annex. First the rules of the buyer agent are presented, and then the rules associated to the seller agent.

The rules assciated to the **ACCEPT** communication primitive of the buyer are first defined.

1) The first **ACCEPT** rule says that an offer for a quantity *q* for a certain product having the price between *minPrice* and *maxPrice*, from an unknown, non-cooperative or slightly cooperative partner is accepted, if the price fulfills different conditions, with respect to the negotiation time elapsed, expressed below;

```
(defrule accept1
  (NegotiationObject {sellerClassification == "nc" ||
sellerClassification == "u" || sellerClassification == "sc"}
                (currentOffer ?o) (quantity ?q) (timeElapsed ?te))
  (Product (minPrice ?minPrice) (maxPrice ?maxPrice))
    => (bind ?half (* ?q (/ (+ ?minPrice ?maxPrice) 2)))
    (bind ?quarter (* ?q (/ (+ ?minPrice ?half) 2)))
    (bind ?test1 (and (<= ?o (* ?q (+ ?minPrice 10))) (<= ?te (* 0.6
?*maxNegTime*))))
    (bind ?test2 (and (<= ?o (* ?quarter ?q)) (and (> ?te (* 0.6
?*maxNegTime*)) (<= ?te (* 0.8 ?*maxNegTime*)))))
    (bind ?test3 (and (<= ?o ( * ?q ?half)) (> ?te (* 0.8
?*maxNegTime*))))
    (if (or ?test1 ?test2 ?test3) then (add (new Offer "accept1"
?*accept*))))
```

2) The second **ACCEPT** rule deals with the case when an offer for a quantity *q* for a certain product having the price between *minPrice* and *maxPrice*, from a cooperative, very cooperative or highly cooperative partner is accepted, if the price fulfills different conditions, with respect to the negotiation time elapsed, expressed below;

```
(defrule accept2
  (NegotiationObject {sellerClassification == "c" ||
sellerClassification == "vc" || sellerClassification == "hc"}
        (currentOffer ?o) (quantity ?q) (timeElapsed ?te))
  (Product (minPrice ?minPrice) (maxPrice ?maxPrice))
    => (bind ?half (* ?q (/ (+ ?minPrice ?maxPrice) 2)))
    (bind ?threeQuarters (* ?q (/ (+ ?maxPrice ?half) 2)))
    (bind ?test1 (and (<= ?o (* ?q (+ ?half 10))) (<= ?te (* 0.6
?*maxNegTime*))))
     (bind ?test2 (and (<= ?o (* ?threeQuarters ?q)) (and (> ?te (* 0.6
?*maxNegTime*)) (<= ?te (* 0.8 ?*maxNegTime*)))))
    (bind ?test3 (and (<= ?o ( * ?q ?maxPrice)) (> ?te (* 0.8
?*maxNegTime*))))
    (if (or ?test1 ?test2 ?test3) then (add (new Offer "accept2"
?*accept*))))
```

For the **REJECT** communication primitive of the buyer, there is the following rule:

1) The buyer will reject an offer for a negotiation object, received after a certain time threshold, which is defined as *3* seconds for each negotiation. The message *Negotiation time elapsed* is sent from the buyer to the seller;

```
(defrule reject1
```

```
      (NegotiationObject (timeElapsed ?te &: (>= ?te ?*maxNegTime*)))
       =>  (add  (new  Offer  "reject1"  ?*reject*  "Negotiation  time
elapsed.")))
```

For the **PROPOSE** communication primitive of the buyer, there are the following rules:

1) The price offered by a buyer, in the beginning, for a certain quantity *q* of products, having the price between *minPrice* and *maxPrice*, to a non-cooperative or unknown seller, is equal to *q*minPrice*;

```
     (defrule firstPrice1
      (NegotiationObject  {step  ==  1}  {sellerClassification  ==  "nc"  ||
sellerClassification == "u"} (quantity ?q))(Product (minPrice ?minPrice))
        => (add (new Offer "firstPrice1" (* ?q ?minPrice) ?*propose*)))
```

2) The price offered by a buyer, in the beginning, for a certain quantity *q* of products, having the price between *minPrice* and *maxPrice*, to a slightly cooperative or cooperative seller, is equal to *q*(minPrice+5)*;

```
     (defrule firstPrice2
      (NegotiationObject  {step  ==  1}  {sellerClassification  ==  "sc"  ||
sellerClassification == "c"} (quantity ?q))(Product (minPrice ?minPrice))
        =>  (add  (new  Offer  "firstPrice2"  (*  ?q  (+  5  ?minPrice))
?*propose*)))
```

3) The price offered by a buyer, in the beginning, for a certain quantity *q* of products, having the price between *minPrice* and *maxPrice*, to a very cooperative or highly cooperative seller, is equal to *q*(minPrice+10)*;

```
     (defrule firstPrice3
      (NegotiationObject  {step  ==  1}  {sellerClassification  ==  "vc"  ||
sellerClassification == "hc"} (quantity ?q))(Product (minPrice ?minPrice))
        =>  (add  (new  Offer  "firstPrice3"  (*  ?q  (+  10  ?minPrice))
?*propose*)))
```

For the **linear strategy** of the buyer, there are the following rules:

1) The buyer will increase its previous offer *o* with *2*, to the non-cooperative or unknown sellers, when the negotiation step is greater than *1*;

```
     (defrule newPrice4
      (NegotiationObject     {step     >     1}     (previousOffer     ?o)
{sellerClassification == "nc" || sellerClassification == "u"}
            (quantity ?q)) (Strategy {strategy == ?*linear*})
      (Product (maxPrice ?maxPrice))
        => (if (> (* ?q ?maxPrice) (+ ?o (* 2 ?q))) then
            (add (new Offer "newPrice4" (+ ?o (* 2 ?q)) ?*propose*))
         else (add (new Offer "newPrice4" (* ?q ?maxPrice) ?*propose*))))
```

2) The buyer will increase its previous offer *o* with *4*, to the slightly cooperative or cooperative sellers, when the negotiation step is greater than *1*;

```
     (defrule newPrice5
      (NegotiationObject     {step     >     1}     (previousOffer     ?o)
{sellerClassification == "sc" || sellerClassification == "c"}
            (quantity ?q)) (Strategy {strategy == ?*linear*})
      (Product (maxPrice ?maxPrice))
        => (if (> (* ?q ?maxPrice) (+ ?o (* 4 ?q))) then
            (add (new Offer "newPrice5" (+ ?o (* 4 ?q)) ?*propose*))
         else (add (new Offer "newPrice5" (* ?q ?maxPrice) ?*propose*))))
```

3) The buyer will increase its previous offer *o* with *6*, to the very cooperative or highly cooperative sellers, when the negotiation step is greater than *1*;

```
     (defrule newPrice6
```

```
      (NegotiationObject     {step     >     1}     (previousOffer     ?o)
{sellerClassification == "vc" || sellerClassification == "hc"}
            (quantity ?q)) (Strategy {strategy == ?*linear*})
      (Product (maxPrice ?maxPrice))
        => (if (> (* ?q ?maxPrice) (+ ?o (* 6 ?q))) then
            (add (new Offer "newPrice6" (+ ?o (* 6 ?q)) ?*propose*))
         else (add (new Offer "newPrice6" (* ?q ?maxPrice) ?*propose*))))
```

For the **conceder strategy** of the buyer, there are the following rules:

1) The buyer will increase its previous offer *o* with *15*, to the non-cooperative or unknown sellers, when the negotiation step is greater than *1* and lower than *12*;

```
     (defrule newPrice7
      (NegotiationObject   {step   >   1   &&   step   <   ?*concederRounds*}
(previousOffer ?o) {buyerClassification == "nc" || buyerClassification ==
"u"}(quantity ?q)) (Product (maxPrice ?maxPrice))
      (Strategy {strategy == ?*conceder*})
        => (if (> (* ?q ?maxPrice) (+ ?o (* 15 ?q))) then
            (add (new Offer "newPrice7" (+ ?o (* 15 ?q)) ?*propose*))
         else (add (new Offer "newPrice7" (* ?q ?maxPrice) ?*propose*))))
```

2) The buyer will increase its previous offer *o* with *20*, to the slightly cooperative or cooperative buyers, when the negotiation step is greater than *1* and lower than *12*;

```
     (defrule newPrice8
      (NegotiationObject   {step   >   1   &&   step   <   ?*concederRounds*}
(previousOffer ?o) {buyerClassification == "sc" || buyerClassification ==
"c"}(quantity ?q)) (Product (maxPrice ?maxPrice))
      (Strategy {strategy == ?*conceder*})
        => (if (> (* ?q ?maxPrice) (+ ?o (* 20 ?q))) then
            (add (new Offer "newPrice8" (+ ?o (* 20 ?q)) ?*propose*))
         else (add (new Offer "newPrice8" (* ?q ?maxPrice) ?*propose*))))
```

3) The buyer will increase its previous offer *o* with *25*, to the very cooperative or highly cooperative sellers, when the negotiation step is greater than *1* and lower than *12*;

```
     (defrule newPrice9
      (NegotiationObject   {step   >   1   &&   step   <   ?*concederRounds*}
(previousOffer ?o) {buyerClassification == "vc" || buyerClassification ==
"hc"}(quantity ?q)) (Product (maxPrice ?maxPrice))
      (Strategy {strategy == ?*conceder*})
        => (if (> (* ?q ?maxPrice) (+ ?o (* 25 ?q))) then
            (add (new Offer "newPrice9" (+ ?o (* 25 ?q)) ?*propose*))
         else (add (new Offer "newPrice9" (* ?q ?maxPrice) ?*propose*))))
```

4) The buyer will increase its previous offer *o* with *2*, to the non-cooperative or unknown sellers, when the negotiation step is greater than *12*;

```
     (defrule newPrice10
      (NegotiationObject  {step  >=  ?*concederRounds*}  (previousOffer  ?o)
{sellerClassification == "nc" || sellerClassification == "u"}
            (quantity ?q)) (Strategy {strategy == ?*conceder*})
      (Product (maxPrice ?maxPrice))
        => (if (> (* ?q ?maxPrice) (+ ?o (* 2 ?q))) then
            (add (new Offer "newPrice10" (+ ?o (* 2 ?q)) ?*propose*))
         else (add (new Offer "newPrice10" (* ?q ?maxPrice) ?*propose*))))
```

5) The buyer will increase its previous offer *o* with *4*, to the slightly cooperative or cooperative sellers, when the negotiation step is greater than *12*;

```
     (defrule newPrice11
```

```
      (NegotiationObject {step >= ?*concederRounds*} (previousOffer ?o)
{sellerClassification == "sc" || sellerClassification == "c"}
            (quantity ?q)) (Strategy {strategy == ?*conceder*})
      (Product (maxPrice ?maxPrice))
        => (if (> (* ?q ?maxPrice) (+ ?o (* 4 ?q))) then
            (add (new Offer "newPrice11" (+ ?o (* 4 ?q)) ?*propose*))
         else (add (new Offer "newPrice11" (* ?q ?maxPrice) ?*propose*))))
```

6) The buyer will increase its previous offer *o* with *6*, to the very cooperative or highly cooperative sellers, when the negotiation step is greater than *12*;

```
     (defrule newPrice12
      (NegotiationObject {step >= ?*concederRounds*} (previousOffer ?o)
{sellerClassification == "vc" || sellerClassification == "hc"}
            (quantity ?q)) (Strategy {strategy == ?*conceder*})
      (Product (maxPrice ?maxPrice))
        => (if (> (* ?q ?maxPrice) (+ ?o (* 6 ?q))) then
            (add (new Offer "newPrice12" (+ ?o (* 6 ?q)) ?*propose*))
         else (add (new Offer "newPrice12" (* ?q ?maxPrice) ?*propose*))))
```

For the **boulware strategy** of the buyer, there are the following rules:

1) The buyer will increase its previous offer *o* with *0.1*, when the negotiation step is greater than *1* and the time elapsed in the negotiation is less than *2* seconds, represented by the global variable *boulwareTime1*;

```
     (defrule newPrice13
      (NegotiationObject {step > 1} (previousOffer ?o) {timeElapsed <
?*boulwareTime1*}(quantity ?q)) (Strategy {strategy == ?*boulware*})
        => (add (new Offer "newPrice13" (+ ?o (* 0.1 ?q)) ?*propose*)))
```

2) The buyer will increase its previous offer *o* with *10*, when the negotiation step is greater than *1* and the time elapsed in the negotiation is greater than *2* seconds, represented by the global variable *boulwareTime1*;

```
     (defrule newPrice14
      (NegotiationObject {step > 1} (previousOffer ?o) {timeElapsed >=
?*boulwareTime1*}(quantity ?q)) (Strategy {strategy == ?*boulware*})
      (Product (maxPrice ?maxPrice))
        => (if (> (* ?q ?maxPrice) (+ ?o (* 10 ?q))) then
            (add (new Offer "newPrice14" (+ ?o (* 10 ?q)) ?*propose*))
         else (add (new Offer "newPrice14" (* ?q ?maxPrice) ?*propose*))))
```

3) The buyer will increase its previous offer *o* with *1000*, when the negotiation step is greater than *1* and the time elapsed in the negotiation is greater than *2.5* seconds, represented by the global variable *boulwareTime2*;

```
     (defrule newPrice15
      (NegotiationObject {step > 1} (previousOffer ?o) {timeElapsed >=
?*boulwareTime2*}(quantity ?q)) (Strategy {strategy == ?*boulware*})
      (Product (maxPrice ?maxPrice))
        => (if (> (* ?q ?maxPrice) (+ ?o (* 20 ?q))) then
            (add (new Offer "newPrice15" (+ ?o (* 20 ?q)) ?*propose*))
         else (add (new Offer "newPrice15" (* ?q ?maxPrice) ?*propose*))))
```

For the seller, regarding the **ACCEPT** communication primitive, there are the following rules:

1) The seller will accept an offer from a non-cooperative or unknown buyer, for a certain quantity of *q* products, having the price between *minPrice* and *maxPrice*, with the price greater than *q*(minPrice+maxPrice)/2*;

```
     (defrule accept1
```

```
      (NegotiationObject      {buyerClassification      ==      "nc"      ||
buyerClassification == "u"} {step > 0}(currentOffer ?o) (quantity ?q))
      (Product (minPrice ?minPrice) (maxPrice ?maxPrice &: (>= ?o (* ?q (/
(+ ?minPrice ?maxPrice) 2))))) =>   (add (new Offer "accept1" ?*accept*)))
```

2) The seller will accept an offer from a slightly cooperative or cooperative buyer, for a certain quantity of *q* products, having the price between *minPrice* and *maxPrice*, with the price between *q*(minPrice+maxPrice)/2* and *q*maxPrice*;

```
     (defrule accept2
      (NegotiationObject      {buyerClassification      ==      "sc"      ||
buyerClassification == "c"} {step > 0}(currentOffer ?o) (quantity ?q))
      (Product (minPrice ?minPrice) (maxPrice ?maxPrice &: (<= ?o (* ?q
?maxPrice)) &: (> ?o (* ?q (/ (+ ?minPrice ?maxPrice) 2)))))
        => (add (new Offer "accept2" ?*accept*)))
```

3) The seller will accept an offer from a very cooperative or highly cooperative buyer, for a certain quantity of *q* products, having the price between *minPrice* and *maxPrice*, with the price lower than or equal to *q*maxPrice*;

```
     (defrule accept3
      (NegotiationObject      {buyerClassification      ==      "vc"      ||
buyerClassification == "hc"} {step > 0}(currentOffer ?o) (quantity ?q))
      (Product (minPrice ?minPrice) (maxPrice ?maxPrice &: (<= ?o (* ?q
?maxPrice)))) =>  (add (new Offer "accept3" ?*accept*)))
```

For the **REJECT** communication primitive of the seller, there are the following rules:

1) The seller will reject an offer for a negotiation object, received after a certain time threshold, which is defined as *3* seconds for each negotiation. The message *Negotiation time elapsed* is sent from the seller to the buyer;

```
     (defrule reject1
      (NegotiationObject (timeElapsed ?te &: (>= ?te ?*maxNegTime*)))
        => (add   (new   Offer   "reject1"   ?*reject*   "Negotiation   time
elapsed.")))
```

For the **PROPOSE** communication primitive of the seller, there are the following rules:

1) The price proposed by a seller, in the beginning, for a certain quantity *q* of products, having the price between *minPrice* and *maxPrice*, to a non-cooperative or unknown buyer, is equal to:

a) *q*maxPrice*, if *q≤10*;

b) *q*maxPrice*0.95*, if *10<q<20*;

c) *q*maxPrice*0.85*, if *q≥20*.

```
     (defrule firstPrice1
      (NegotiationObject  {step  ==  0}  {buyerClassification  ==  "nc"  ||
buyerClassification == "u"} (quantity ?q))(Product (maxPrice ?maxPrice))
        => (if (and (> ?q 10) (< ?q 20)) then
           (add  (new  Offer  "firstPrice1"  (*  (*  ?maxPrice  0.95)  ?q)
?*propose*)) elif (>= ?q 20) then
           (add  (new  Offer  "firstPrice1"  (*  (*  ?maxPrice  0.85)  ?q)
?*propose*)) else
           (add (new Offer "firstPrice1" (* ?maxPrice ?q) ?*propose*))))
```

2) The price proposed by a seller, in the beginning, for a certain quantity *q* of products, having the price between *minPrice* and *maxPrice*, to a slightly cooperative or cooperative buyer, is equal to:

a) *q*maxPrice*0.97*, if *q≤10*;

b) *q*maxPrice*0.9*, if *10<q<20*;

c) *q*maxPrice*0.8*, if *q≥20*.

```
     (defrule firstPrice2
```

```
      (NegotiationObject {step == 0} {buyerClassification == "sc" ||
buyerClassification == "c"} (quantity ?q))(Product (maxPrice ?maxPrice)  )
        => (if (and (> ?q 10) (< ?q 20)) then
            (add (new Offer "firstPrice2" (* (* ?maxPrice 0.90) ?q)
?*propose*)) elif (> ?q 20) then
            (add (new Offer "firstPrice2" (* (* ?maxPrice 0.80) ?q)
?*propose*)) else
            (add (new Offer "firstPrice2" (* (* ?maxPrice 0.97) ?q)
?*propose*))))
```

3) The price proposed by a seller, in the beginning, for a certain quantity *q* of products, having the price between *minPrice* and *maxPrice*, to a very cooperative or highly cooperative buyer, is equal to:

a) *q*maxPrice*0.95*, if *q≤10*;

b) *q*maxPrice*0.85*, if *10<q<20*;

c) *q*maxPrice*0.75*, if *q≥20*.

```
     (defrule firstPrice3
      (NegotiationObject {step == 0} {buyerClassification == "vc" ||
buyerClassification == "hc"} (quantity ?q))(Product (maxPrice ?maxPrice)  )
        => (if (and (> ?q 10) (< ?q 20)) then
            (add (new Offer "firstPrice3" (* (* ?maxPrice 0.85) ?q)
?*propose*)) elif (> ?q 20) then
            (add (new Offer "firstPrice3" (* (* ?maxPrice 0.75) ?q)
?*propose*)) else
            (add (new Offer "firstPrice3" (* (* ?maxPrice 0.95) ?q)
?*propose*))))
```

For the **linear strategy** of the seller, there are the following rules:

1) The seller will decrease its previous offer *o* with *2*, to the non-cooperative or unknown buyers, when the negotiation step is greater than *1*;

```
     (defrule newPrice4
      (NegotiationObject {step > 1} (previousOffer ?o) {buyerClassification
== "nc" || buyerClassification == "u"} (quantity ?q)) (Product (minPrice
?minPrice)) (Strategy {strategy == ?*linear*})
        => (if (<= (* ?q ?minPrice) (- ?o (* 2 ?q))) then
            (add (new Offer "newPrice4" (- ?o (* 2 ?q)) ?*propose*))
         else (add (new Offer "newPrice4" (* ?q ?minPrice) ?*propose*))))
```

2) The seller will decrease its previous offer *o* with *4*, to the slightly cooperative or cooperative buyers, when the negotiation step is greater than *1*;

```
     (defrule newPrice5
      (NegotiationObject {step > 1} (previousOffer ?o) {buyerClassification
== "sc" || buyerClassification == "c"}(quantity ?q)) (Product (minPrice
?minPrice))(Strategy {strategy == ?*linear*})
        => (if (<= (* ?q ?minPrice) (- ?o (* 4 ?q))) then
            (add (new Offer "newPrice5" (- ?o (* 4 ?q)) ?*propose*))
         else (add (new Offer "newPrice5" (* ?q ?minPrice) ?*propose*))))
```

3) The seller will decrease its previous offer *o* with *6*, to the very cooperative or highly cooperative buyers, when the negotiation step is greater than *1*;

```
     (defrule newPrice6
      (NegotiationObject {step > 1} (previousOffer ?o) {buyerClassification
== "vc" || buyerClassification == "hc"}(quantity ?q)) (Product (minPrice
?minPrice))(Strategy {strategy == ?*linear*})
        => (if (<= (* ?q ?minPrice) (- ?o (* 6 ?q))) then
```

```
          (add (new Offer "newPrice6" (- ?o (* 6 ?q)) ?*propose*))
        else (add (new Offer "newPrice6" (* ?q ?minPrice) ?*propose*))))
```

For the **conceder strategy** of the seller, there are the following rules:

1) The seller will decrease its previous offer *o* with *15*, to the non-cooperative or unknown buyers, when the negotiation step is greater than *1* and lower than *12*;

```
(defrule newPrice7
  (NegotiationObject {step > 1 && step < ?*concederRounds*}
(previousOffer ?o) {buyerClassification == "nc" || buyerClassification ==
"u"}(quantity ?q)) (Product (minPrice ?minPrice))
(Strategy {strategy == ?*conceder*})
    => (if (<= (* ?q ?minPrice) (- ?o (* 15 ?q))) then
        (add (new Offer "newPrice7" (- ?o (* 15 ?q)) ?*propose*))
      else (add (new Offer "newPrice7" (* ?q ?minPrice) ?*propose*))))
```

2) The seller will decrease its previous offer *o* with *20*, to the slightly cooperative or cooperative buyers, when the negotiation step is greater than *1* and lower than *12*;

```
(defrule newPrice8
  (NegotiationObject {step > 1 && step < ?*concederRounds*}
(previousOffer ?o) {buyerClassification == "sc" || buyerClassification ==
"c"}(quantity ?q)) (Product (minPrice ?minPrice))
  (Strategy {strategy == ?*conceder*})
    => (if (<= (* ?q ?minPrice) (- ?o (* 20 ?q))) then
        (add (new Offer "newPrice8" (- ?o (* 20 ?q)) ?*propose*))
      else (add (new Offer "newPrice8" (* ?q ?minPrice) ?*propose*))))
```

3) The seller will decrease its previous offer *o* with *25*, to the very cooperative or highly cooperative buyers, when the negotiation step is greater than *1* and lower than *12*;

```
(defrule newPrice9
  (NegotiationObject {step > 1 && step < ?*concederRounds*}
(previousOffer ?o) {buyerClassification == "vc" || buyerClassification ==
"hc"}(quantity ?q)) (Product (minPrice ?minPrice))
  (Strategy {strategy == ?*conceder*})
    => (if (<= (* ?q ?minPrice) (- ?o (* 25 ?q))) then
        (add (new Offer "newPrice9" (- ?o (* 25 ?q)) ?*propose*))
      else (add (new Offer "newPrice9" (* ?q ?minPrice) ?*propose*))))
```

4) The seller will decrease its previous offer *o* with *q*2*, to the non-cooperative or unknown buyers, when the negotiation step is greater than *12*;

```
(defrule newPrice10
  (NegotiationObject {step >= ?*concederRounds*} (previousOffer ?o)
{buyerClassification == "nc" || buyerClassification == "u"}
        (quantity ?q)) (Product (minPrice ?minPrice))
  (Strategy {strategy == ?*conceder*})
    => (if (<= (* ?q ?minPrice) (- ?o (* 2 ?q))) then
        (add (new Offer "newPrice10" (- ?o (* 2 ?q)) ?*propose*))
      else (add (new Offer "newPrice10" (* ?q ?minPrice) ?*propose*))))
```

5) The seller will decrease its previous offer *o* with *q*4*, to the slightly cooperative or cooperative buyers, when the negotiation step is greater than *12*;

```
(defrule newPrice11
  (NegotiationObject {step >= ?*concederRounds*} (previousOffer ?o)
{buyerClassification == "sc" || buyerClassification == "c"}(quantity ?q))
(Product (minPrice ?minPrice))(Strategy {strategy == ?*conceder*})
    => (if (<= (* ?q ?minPrice) (- ?o (* 4 ?q))) then
        (add (new Offer "newPrice11" (- ?o (* 4 ?q)) ?*propose*))
```

```
        else (add (new Offer "newPrice11" (* ?q ?minPrice) ?*propose*))))
```

6) The seller will decrease its previous offer *o* with *q*6*, to the very cooperative or highly cooperative buyers, when the negotiation step is greater than *12*;

```
    (defrule newPrice12
     (NegotiationObject {step >= ?*concederRounds*} (previousOffer ?o)
{buyerClassification == "vc" || buyerClassification == "hc"}(quantity ?q))
(Product (minPrice ?minPrice))(Strategy {strategy == ?*conceder*})
        => (if (<= (* ?q ?minPrice) (- ?o (* 6 ?q))) then
           (add (new Offer "newPrice12" (- ?o (* 6 ?q)) ?*propose*))
         else (add (new Offer "newPrice12" (* ?q ?minPrice) ?*propose*))))
```

For the **boulware strategy** of the seller, there are the following rules:

1) The seller will decrease its previous offer *o* by *0.1*q*, when the negotiation step is greater than *1* and the time elapsed in the negotiation is less than *2* seconds, represented by the global variable *boulwareTime1*;

```
    (defrule newPrice13
     (NegotiationObject {step > 1} (previousOffer ?o) {timeElapsed <
?*boulwareTime1*}(quantity ?q))(Strategy {strategy == ?*boulware*})
        => (add (new Offer "newPrice13" (- ?o (* ?q 0.1)) ?*propose*)))
```

2) The seller will decrease its previous offer *o* by *10*q*, when the negotiation step is greater than *1* and the time elapsed in the negotiation is greater than or equal to *2* seconds, represented by the global variable *boulwareTime1*;

```
    (defrule newPrice14
     (NegotiationObject {step > 1} (previousOffer ?o) {timeElapsed >=
?*boulwareTime1*}(quantity ?q)) (Product (minPrice ?minPrice))
     (Strategy {strategy == ?*boulware*})
        => (if (<= (* ?q ?minPrice) (- ?o (* 10 ?q))) then
           (add (new Offer "newPrice14" (- ?o (* 10 ?q)) ?*propose*))
         else (add (new Offer "newPrice14" (* ?q ?minPrice) ?*propose*))))
```

3) The seller will decrease its previous offer *o* by *20*q*, when the negotiation step is greater than *1* and the time elapsed in the negotiation is greater than or equal to *2.5* seconds, represented by the global variable *boulwareTime2*;

```
    (defrule newPrice15
     (NegotiationObject {step > 1} (previousOffer ?o) {timeElapsed >=
?*boulwareTime2*}(quantity ?q))(Strategy {strategy == ?*boulware*})
     (Product (minPrice ?minPrice))
        => (if (<= (* ?q ?minPrice) (- ?o (* 20 ?q))) then
           (add (new Offer "newPrice15" (- ?o (* 20 ?q)) ?*propose*))
         else (add (new Offer "newPrice15" (* ?q ?minPrice) ?*propose*))))
```