ISCTE IUL
Instituto Universitário de Lisboa

Department of Information Science and Technology

School of Technology and Architecture

Prenegotiation and Actual Negotiation in Electricity Markets

By
Hugo Gonçalo Leonardo Lopes

A Dissertation presented in partial fulfillment of the Requirements for the Degree of Master in Computer Science Engineering - specialization in Information Systems and Knowledge Management

Supervisors:
Doutor Fernando Jorge Ferreira Lopes, LNEG

Prof. Nuno Manuel Mendes da Cruz David, ISCTE

September, 2012

Acknowledgments

I would like to express my gratitude to my supervisor, Fernando Lopes, Ph.D., and to my professor and co-supervisor, Nuno David, Ph.D., for the opportunity to work on this dissertation, and for the continuous support throughout the last year.

My parents, João Duarte and Manuela Lopes, for prioritizing and supporting mine and my sister's education. To my sister, Cláudia Lopes, for helping me any time I needed it.

For my friends and colleagues, in particular, Gonçalo Carvalho, Marcos Pinto, Tiago Martinho, Renato Martins, and Henrique Graça, for the many hours put on academic projects and helpful discussions.

To ISCTE (Instituto Universitário de Lisboa), including the professors, whose motivation and effort allowed me to reach this stage.

Finally, Laboratório Nacional de Energia e Geologia (LNEG), for providing me an environment to work on my dissertation, and to Cristina Ilco and João Filipe, who there helped me better understand the Energy Market.

Abstract

Software agents have been successfully used in a vast range of applications. Agents have been gradually designed to act in open environments and to manage their cooperative and competitive interactions with other agents present in their environment. In a Multi-agent System (MAS), involving different agents operating individually to meet their design goals, conflict will be inevitable — it is not necessarily bad or good, but it is inevitable. Conflict is the focal point of interaction, i.e. the driving force of negotiation. Furthermore, conflict is the element that connects the individual and social behavior of agents.

Software tools based on intelligent agents with negotiating capabilities have became important and pervasive. Particularly, there is a growing demand to develop MAS featuring bilateral contracts in liberalized Electricity Markets (EMs). This dissertation addresses, at least in part, this challenge by presenting the computational tool NSEM – Negotiation Simulator for Electricity Markets. NSEM features Belief-Desire-Intention agents able to effectively plan actions, manage conflicts, and trade proposals to reach mutually beneficial agreements.

NSEM focuses on the preliminary activities that should come before negotiation, usually referred to as prenegotiation. These activities include the definition of the issues at stake, their prioritization, and the selection of an appropriate protocol and effective strategies. This dissertation presents details of NSEM's implementation and test. NSEM was developed with the JAVA programming language and the JADE platform. Its test was performed by using a case study, featuring prenegotiation and actual negotiation of bilateral contracts in liberalized EMs.

Keywords

Autonomous Agents; Conflict of Interests; Automated Negotiation; Prenegotiation; Electricity Markets

Resumo

A tecnologia baseada em agentes computacionais autónomos tem vindo a ser utilizada com sucesso numa vasta gama de aplicações. Num Sistema Multi-agente (SMA), composto por diversos agentes atuando individualmente para alcançar os seus objectivos de projecto, os conflitos são inevitáveis. Os conflitos constituem o elemento que liga o comportamento individual e social dos agentes, sendo normalmente a força motriz da negociação.

O desenvolvimento de agentes com capacidade negocial sofreu avanços significativos ao longo dos últimos anos. Estes agentes apresentam diversas vantagens relativamente aos negociadores humanos, sendo de realçar a capacidade de obterem acordos benéficos para todas as partes envolvidas na negociação. Nesta perspectiva, salienta-se a procura crescente de SMAs para simular a contratação bilateral de energia em mercados liberalizados.

Esta dissertação tenta responder a este desafio através do desenvolvimento da ferramenta computacional NSEM: "Negotiation Simulator for Electricity Markets". NSEM permite criar agentes constituídos pelas atitudes mentais de crença, desejo e intenção, capazes de planear ações de forma efectiva, gerir conflitos, e negociar acordos mutualmente beneficiáveis.

NSEM coloca a ênfase no conjunto de atividades preliminares a realizar antes da negociação, referido usualmente como pré-negociação. Estas atividades incluem a definição dos itens a negociar, as suas prioridades, a escolha de um protocolo apropriado, e a seleção de estratégias efetivas. Esta dissertação apresenta detalhes da implementação e teste do NSEM. A implementação foi efetuada através do Java e do JADE. O teste foi realizado através do desenvolvimento de um caso de estudo referente à contratação bilateral de energia em mercados de eletricidade liberalizados.

Palavras Chave

Agentes Autónomos; Conflito de interesses; Negociacão automática; Pré-negociacão; Mercado de Eletricidade

Contents

List of Figures

List of Tables

Abbreviations

MAS Multi-agent System

EM Electricity Market

BDI Belief-Desire-Intention

SMP System Marginal Price

EMCAS Electricity Market Complex Adaptive System

REN Redes Energéticas Nacionais

MIBEL Iberian Electricity Market

SEPIA Simulator for the Electric Power Industry Agents

AMES Agent Based Modelling of Electricity Systems

MASCEM Multi-Agent System that Simulates Competitive Electricity Markets

OAA Open Agent Architecture

ICL Interagent Communication Language

Repast S Repast Simphony

JADE Java Agent Development Framework

API Application programming interface

IS Intention Structure

FIPA Foundation for Intelligent Physical Agents

ACL Agent Communication Language

SME Small and Medium Enterprises

XML Extensible Markup Language

GUI Graphical User Interface

OMEL Operador del Mercado Ibérico de Energía

NYSEG New York State Electric & Gas

NSEM Negotiation Simulator for the Energy Market

LNEG Laboratório Nacional de Energia e Geologia

List of Symbols

S	Set of environment states	11
A	Set of actions	11
ag_i	An intelligent agent	14
ag_j	An intelligent agent different from ag_i	15
Ag	A set of autonomous agents	27
ag_i	An agent $\in Ag$	27
B_i	A set of beliefs possessed by ag_i	27
b_{im}	A belief possessed by ag_i	27
bh_{im}	The header of b_{im}	27
bb_{im}	The body of b_{im}	27
bd_{im}	Description of the belief b_{im}	27
G_i	A set of goals ag_i wants to achieve	27
g_{ik}	A goal possessed by $ag_i and \in G_i$	27
gb_{ik}	A statement that identifies g_{ik}	27
PL_i	A library of plan templates from ag_i	27
pt_{iz}	A plan template $\in PL_i$	27
pth_{iz}	Header of plan template pt_{iz}	27
ptt_{iz}	The type of pt_{iz}	27
$ptpc_{iz}$	The preconditions that must be true before pt_{ij} can executed	27
ptb_{iz}	The body of the plan template pt_{iz}	27
pte_{iz}	describes a set of statements that should be hold after pt_{iz}	27
ptn_{iz}	Name of pt_{iz}	28
pta_{iz}	A set of arguments for ptn_{iz}	28
APT_{iy}	Applicable plan templates for g_{iy}	28
$AltP_{iy}$	A set of alternative plans for goal g_{iy}	28
IS_i	Intention structure of ag_i	29
int_{jk}	An intention $\in IS_j$ from ag_j	29
IS_j	Intention structure of ag_j	29
int_{jk}	An intention $\in IS_j$ from ag_j	29
$Conf_{i,j}$	Conflict between agi and ag_j	29

$Incomp$	Incompatibility between intentions	29
PP_i	A set of possible plans ag_i believes agents in its environment have generated	29
PI_i	A set of possible intentions ag_i believes agents in its environment have generated	29
$PotConf_{i,j}$	Potential Conflict between agi and ag_j	30
$int_{im}(ag_j)$	Intention $int_{im}(ag_j) \in PI_i$ an represents an intention ag_i believes ag_j has	30
$prop^t_{sender,receiver}$	proposal sent on step t from $sender$ to $receiver$	32
t	Temporal step of the negotiation	32
I	Set of issues included in each proposal	34
R(M)	Random value within [0,M]	34
$weight_{ix_n}$	Weight attributed to x_n	34

1 Introduction

Contents

1.1 Background

Software agents, also known as intelligent agents, have been successfully used in a vast range of applications. From industrial to medical, they have changed the way developers approach software design. In the last few years, agents have begun to gradually be designed to act in open environments, with incomplete and uncertain information, limited resources, and able to manage their cooperative and competitive interactions with the other agents. The keyword became "interaction"– between an agent and its environment, between agents operating in the same environment, or even between agents and humans [1].

Industrial applications were the first type of applications to exploit the features of software agents, using them in process management, telecommunications, air traffic control and transportation. There are also numerous commercial applications, ranging from information management and electronic commerce, to business process management. Regarding the medical field, software agents are found in patient monitoring applications, generic medical treatments, and in many others life-depending applications.

Multi-agent Systems (MAS) are systems composed of multiple agents that interact to solve problems that are beyond the individual capabilities of each agent. They offer modularity. Accordingly, if a problem domain is particularly complex, large, or unpredictable, a good way to solve it is to develop a number of agents functionally specific to solve particular problems present in that problem domain [2]. An area that reflects these problems, and can benefit from a Multi-agent System (MAS) approach is the liberalized Electricity Market (EM).

Generally speaking, the electricity sector has four large areas of activity: production, transportation, distribution and commercialization. The electricity is generated in power plants and transported in high voltage through a transmission grid to electric substations near residential areas, or other demand areas, where it is commercialized. The liberalized EM, introduced in the 1980s in some countries, transformed what was a monopoly into a sector with the functions of electricity production and commercialization separated from the functions of transportation and distribution. This detachment created two markets, a wholesale market, where diverse producers competitively sell electricity to retailers, and a retail market, where retailers competitively resell the electricity to end consumers. Accordingly, this considers a MAS involving the following types of agents:

1. Producers or generators, which represent generation companies that sell electricity to a wholesale market;

2. Retailers or suppliers, representing retail companies that buy electricity from a wholesale market and sell it in a retail market;

3. Consumers or clients, which represent end consumers that buy electricity in a retail market.

Commercialization of electricity in the EM may be performed in pools, by bilateral contracts, or through hybrid models (involving aspects of pools and bilateral contracts). Pool markets are a form of auction, where participants can send bids to sell and buy electricity, for a certain period of time, to an entity referred to as market operator, who in turn analyzes all the bids submitted and calculates a market price that must be followed by all participants. On the other hand, bilateral contracts involve basically the negotiation of prices, volumes, periods, among other possible issues, between two traders.

Multi-agent systems allow for simulation and analysis of real-life complex systems. The agent-based view provides tools and techniques that can assist the construction, implementation and study of both small and large computer systems [3]. Conceptually, a MAS presents itself as a good way to represent distributed domains such as the EM, and to simulate the interactions and dynamics between the different entities participating in it. However, when the information and/or control is distributed as it is in a MAS, a set of problems are raised related to the design and effective operation of the system, such as [3, 4]:

1. The design problem – how to formulate, describe, decompose, and allocate problems and synthesize results among a group of intelligent agents?

2. The coordination problem – how to ensure that agents act coherently in making decisions or taking action, accommodating the non-local effects of local decisions and avoiding harmful interactions?

3. The discord problem – How to recognize and reconcile disparate viewpoints and conflicting intentions among a collection of agents trying to coordinate their actions?

The design problem is related to the domain in which the agents operate, how that domain is represented, and how the domain problems are addressed. In this dissertation, the domain is the liberalized EM, in particular, the commercialization of electricity through bilateral contracts.

The coordination and discord problems are related to the way agents react to social conflicts, because in a world with multiple agents, each trying to meet its objectives in an autonomous manner, conflicts will inevitably surface. These conflicts can be resolved through negotiation.

Negotiation is an important form of social interaction, composed of different phases (and processes), notably prenegotiation, actual negotiation, and deal execution. Prenegotiation is the main focus of the work in this dissertation. It involves mainly the construction of a plan accounting for the activities that negotiators should attend to before the negotiation process begins. Typically, these activities include:

1. Defining the negotiation issues, i.e. the negotiation agenda;

2. Defining targets and limits for each issue;

3. Selecting an appropriate protocol;
4. Selecting a negotiation strategy.

1.2 Motivation and Research Questions

The research of autonomous agents with negotiating capabilities has increased over the last years. Generally speaking, researchers have adopted two distinct approaches to the design of the agents [5]:

Theoretical or Formal – based on agent specification; involves the development of formal models of autonomous agents with negotiating capabilities, which typically are not associated to any computational system;

Practical or Computational – based on agent implementation; involves the development of computational models and their experimental validation; these models define data structures, the processes that manage those structures, and the flow of information between various processes;

This dissertation adopts a practical approach. In the last years, a number of software tools based on intelligent agents with negotiating capabilities have been introduced to model liberalized markets. However, in spite of the capabilities and versatility of existing systems, most of them are limited to particular domains and specific types of agents. Presently, there is a growing demand to develop MAS featuring bilateral contracts in EMs.

When considering the aforementioned negotiation systems, it's clear where the research focus has been put on, namely on the negotiation process itself, i.e. the attempt to acquire a favorable agreement. The preliminary activities that precede the negotiation, i.e. the prenegotiation process, have not received much attention [6].

As mentioned above in section 1.1, negotiation has its foundation on social conflict. However, most existing computational models and frameworks do not consider the origins, causes and motivations of conflict. Furthermore, it is legitimate to create a generic negotiation model that ignores a specific agent architecture, i.e. an agent that emphasizes the social behavior while ignoring the individual behavior. This allows for the latter integration of many types of agent's architectures, but raises the question on how to integrate them:

1. *How to integrate an autonomous agent's individual behavior (in particular, the capability to plan actions) with its social behavior (the capability to negotiate contracts) ?*

The key for a successful negotiation is in the planning and preparation that precedes the negotiation. This is in accordance with successful human negotiators. While persuasive arguments, astute communications skills, and clever maneuvers may be important, they do not overcome the disadvantages brought on by bad planning and preparation. The foundation for success is in the preparation and planning realized before the negotiation itself [6]. Accordingly, we highlight the following question:

2. *Successful human negotiators often consider prenegotiation, i.e. planning and preparation for negotiation, a crucial factor for a successful negotiation outcome. How to develop software agents able to effectively plan and prepare for a negotiation?*

Existing EM simulators include agents with limited capabilities in planning for negotiation. The focus is on to the negotiation process. Little attention is given to the identification of the negotiation issues and the resulting negotiation agenda [6]. In this dissertation, the aim is to create agents capable of preparing (efficiently) for negotiation, and to equip them with social aspects, applying the Java programming language and the Java Agent Development Framework (JADE) platform [1]. We highlight the following issue:

3. *How to develop software agents capable of negotiating bilateral contracts in the EM, with prenegotiation on its foundation?*

1.3 Objectives

The main objectives of this dissertation are detailed as follows:

1. To develop a simplified EM, composed of autonomous agents representing electricity producers, retailers and end consumers; the focus will be on the retail market;
2. To implement simplified Belief-Desire-Intention (BDI) agents capable of creating plans to meet their design objectives; furthermore, the agents should be able to detect conflicts involving other agents present in the environment;
3. Implement autonomous agents with the capabilitiy to negotiate bilateral contracts; in particular, the dissertation focuses on the development of agents capable to effectively plan for negotiation (prenegotiation) and able to exchange negotiation proposals and counterproposals in an iterative way;
4. To develop and implement a case study relative to a retail market, involving negotiation through bilateral contracts, between a retail representative and a electricity consumer, and focusing on the negotiation preparation.

[1]This work was performed under the project MAN-REM: Multi-agent Negotiation and Risk Management in Electricity Markets (FCOMP-01-0124-FEDER-020397), and supported by both FEDER and National funds through the program “COMPETE—Programa Operacional Temático Factores de Competividade.

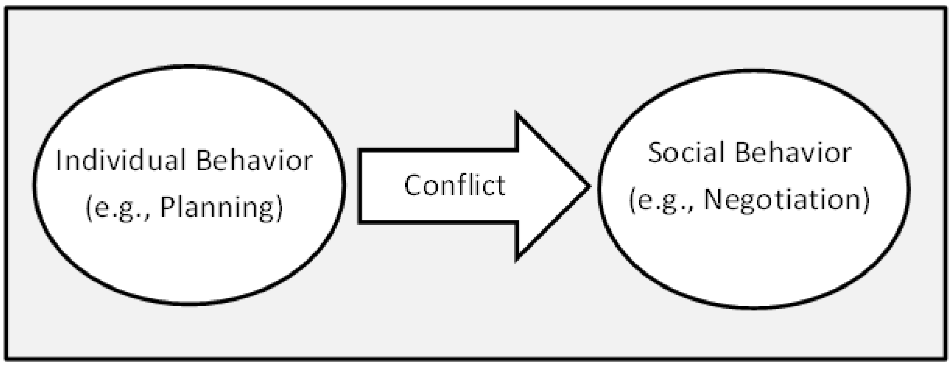


Figure 1.1: Conflict as the foundation for the Individual-Social behavior connection

All the objectives above are accomplished by utilizing the tools and techniques of MASs. Furthermore, the approach adopted in this dissertation is shown in figure 1.1: the conflict, as the key concept, will tie-in individual and social aspects of an agent behaviors.

1.4 Structure

This dissertation is composed by six chapters and two appendices:

1. A succinct introduction to the background of this dissertation, the motivation, and the main objectives.

2. The State of the Art. In this chapter, crucial topics are discussed as they were presented previously in existing publications. It starts with an attempt to find different definitions of autonomous agents, a discussion of their properties and environments. The various types of agents are shortly described, including the BDI architecture, that will be featured in the developed system. Conflict of interests is then presented from the point of view of five authors with different approaches, including the one adopted in the dissertation. Next, the different facets of negotiation are described, its different properties and processes, and the possible agent development approaches. The EM's composition is elaborated on and its liberalization contextualized. Various existing simulators for the EM are detailed and analyzed side by side. To end the chapter, some platforms capable of assisting MAS development are described.

3. The third chapter will share some topics with the previous chapter, but now the concepts are described as they were implemented in the developed system. It starts with an in depth description of the agent's architecture, notably, the plan generation process. It is followed by a definition of conflict, how it is implemented and how it relates to the overall system. The chapter concludes by detailing the various steps that need to be taken in order to model the prenegotiation and execute the negotiation.

4. The fourth chapter, we present a more technical look at the simulator developed, Negotiation Simulator for the Energy Market (NSEM), and its implementation. The flow of execution for each main agent type is presented, their protocols for communication are detailed, and a small simulator walkthrough is done for an easier user experience, using the Graphical User Interface (GUI).

5. A case study using NSEM that considers the prenegotiation and actual negotiation processes, in a retail market of the EM.

6. The conclusions. It includes a summation of the work performed in this dissertation, a discussion of the objectives reached, the dissertation's contributions, and suggestions for future work.

7. The first appendix presents a few methods from NSEM in the Java language.

8. The second appendix presents partial data files from NSEM, in Extensible Markup Language (XML).

2

Autonomous Agents, Automated Negotiation, and Energy Markets: A State of Art

Contents

2.1 Autonomous Agents

There is no universal consensus for the definition of the term “agent”, but it is widely accepted that autonomy is at its core. Wooldridge [7] presents an adapted version of a pre-existing definition for an agent: “An Agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives”.

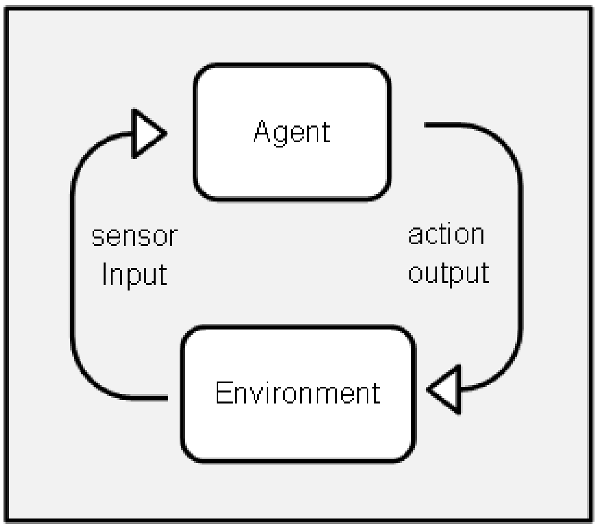


Figure 2.1: An agent interacting with and environment

Another definition, presented by Russel and Norvig [8], and illustrated in figure 2.1, is as follows: ”An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.”.

Jennings and Wooldridge [9] distinguishes two usages for the term agent. The first being weak and relatively uncontentious, and the second strong and potentially more contentious. The week notion is used to describe hardware or software that denotes the following properties:

- *Autonomy*: agents operate without direct intervention, human or other, and have some kind of control over their actions and internal state;
- *Social ability*: agents exhibit some kind of language, which can be used to interact with other agents and possibly humans;
- *Reactivity*: agents perceive their environment and use the perceived information to respond in a timely manner to changes that occur in that environment;
- *Pro-activeness*: agents do not simply act in response to their environment, they are able to exhibit goal-directed behavior by taking the initiative.

The strong notion of agency is utilized by some researchers to denote an agent that shows the proprieties described above, and is also implemented using mental attitudes that are generally associated with humans, like knowledge, belief, intention and obligation.

The literature commonly distinguishes different types of agents, notably [9]: (*i*) purely reactive agents, (*ii*) deliberative agents, (*iii*) hybrid agents, and (*iv*) bdi agents. A brief description of each type follows.

Purely Reactive Agents

Purely reactive agents decide what actions to take without taking into account their history. They base the decision making process only on the present. Putting it simply, they just react to changes that occur in the environment. This process is represented as follows [7]:

$$action : \mathrm{S} \rightarrow A$$

where:

(i) $S = \{s_1, s_2, ...\}$ is a set of *environment states*;

(ii) $A = \{a_1, a_2, ...\}$ a set of actions.

Deliberative Agents

These agents feature an explicit symbolic model of the environment where they operate in. As illustrated in figure 2.2, for each new data perception, an agent updates its information about the environment, and plans the actions necessary to meet its design objectives based on that information.

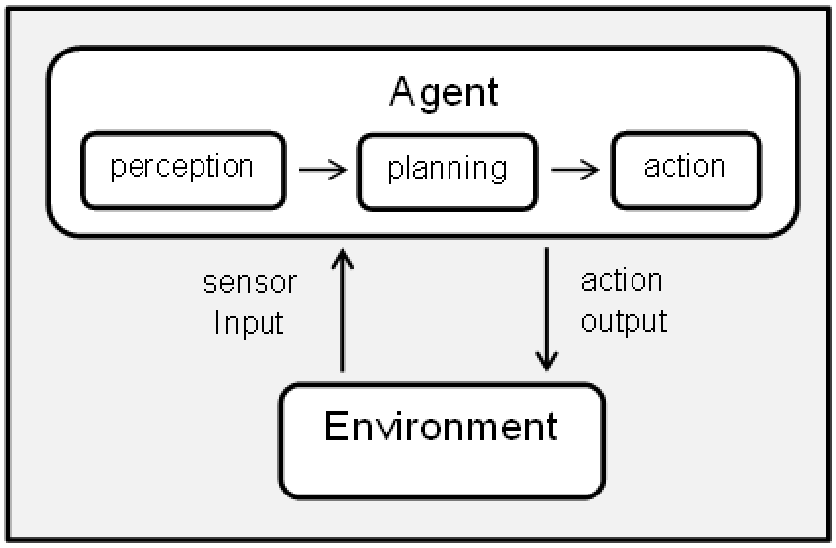


Figure 2.2: The innerworks of a deliberative agent

BDI Agents

Belief-Desire-Intention (BDI) architectures have their roots on the practical reasoning [10], which is a process founded in the philosophical realms denoting the importance of continuously deciding which actions to perform towards achieving predefined goals. The process itself requires two sub-processes[7]:

- deliberation: deciding *what* goals we, as humans, want to achieve;
- means-ends: determining *how* to achieve these goals.

This process is adopted by BDI agents, which have been successfully applied to various applications [11]. The decision process begins by identifying what *options* are available. After having this set of options, also known as alternatives, agents must *commit* to some of them, which are referred to as *intentions*. For each intention, agents should make a *reasonable attempt* to achieve it. This means [7]:

- Agents must carry out a course of action that they *believe* will lead to a path that will satisfy an intention;
- Agents will not entertain options that are inconsistent with previously adopted intentions;
- Agents have to *persist* with an adopted intention and attempt to achieve it; however, if it becomes clear that they will never achieve it, they should drop it;
- If the reason for having an intention disappears, the intention should be dropped;
- If agents adopt an intention, they must believe there is a good chance to achieve it.

It is clear that sometimes agents should drop an intention, making it worthwhile to stop and reconsider the adopted plans. The key issue is when should an agent stop to reconsider adopted plans (or intentions). Reconsideration comes at a cost, both in time and computer resources, so there is a need to decide how often it occurs. The decision should be based on the environment where the agents are located in. If the environment does not change quickly, then it is prone to a more pro-active (goal directed) behavior. However, if the environment changes frequently, reactive (event driven) agents show frequently a better performance than pro-active agents [7].

Hybrid Agents

This architecture results from the combination of features from both deliberative and reactive agents. It tries to overcome disadvantages from both architectures. More specifically, on the deliberative side, its important to stress the incapability to react immediately to changes that occur in the environment, and on the reactive side, the incapability to rationalize decision making when trying to efficiently achieve design goals. The resulting architecture is composed of different hierarchical layers, with reactive layers having priority over deliberative layers, much to assure immediate responses to changes that occur in the environment.

Agent Environments

Russel and Norvig [8] identified a number of dimensions to categorize environments. Some of these dimensions follows:

- *Fully observable vs. partially observable*: When at each point in time, an agent has access through its sensors to the state of the environment it is located in, that environment is considered fully observable. An environment may be considered partially observable as a result of noisy and inaccurate sensors or missing elements from the sensor data, such as information not privy to one agent;
- *Deterministic vs. stochastic*: If the next state of the environment is ascertained exclusively by the current state and the actions performed by the agent, the environment is considered deterministic; otherwise, it is regarded as stochastic;
- *Episodic vs. sequential*: In episodic environments, the agent's experience is divided in episodes, each consisting of a perception followed by a single action. The choice of action does not depend on actions taken in previous episodes, only on the current episode itself. In sequential environments, a current decision could affect all future decisions;
- *Static vs. dynamic*: Dynamic environments may change while an agent is deciding what action to perform. The environment changes with the passage of time, and if the agent does not perform an action within the allotted time, it considers the agent as taking no action. In static environments, neither the environment nor the agent accounts for the passage of time and the next state comes only after the agent has decided what to do. There also a semi-dynamic environment. In this case, the environment does not consider the time, but the agent does, affecting its own performance score;
- *Discrete vs. continuous*: A discrete-state environment has a limited number of distinct states, and the same notion is applied to both the way the time is handled and the perceptions and actions of the agent;
- *Single agent vs. multiagent*: Single agent environments are environments where only one agent is present. Depending on the dynamic between the agents, a multiagent environment may be viewed as either a competitive environment, a cooperative environment, or a combination of both.

2.2 Conflict of interests

Consider the following two main components of an autonomous agent operating in a multiagent system [5]:

- Individual component, which refers to the internal rationalization drawn by an agent;
- Social component, defined by the interaction and communication between distinct agents.

These components are brought together by the presence of conflict. Before we continue delineating agent conflict, let us first look at human conflict. Julia Galliers [12] has identified several main attributes in human conflict, notably:

1. Each social conflict requires, at least, two parties;
2. Each conflict has important purposes and consequences;
3. Opposing values, exclusive values, and/or incompatible values are fundamental facets of each conflict, and these values should be a result of limited resources (symbolic or material).

Julia Galliers has then developed a definition of agent conflict based on their features. She states that conflict consists in a relationship between two agents, in respect to one proposition:

"Conflict of goals or conflict of beliefs exist between one agent and another, when the agents' beliefs or goals with respect to the same proposition [p] are believed by the one agent to be in opposition, and this agent also has a persistent goal to change the other's belief or goal...''

Conte and Castelfranchi [1] defined conflict based on the concepts of interest and counter-interest:

"A conflict of interest between two agents [x and y] with regard to a given [world] state q occurs when q is an interest of x's goal that [a world state] p be true at time t, while it is a counter-interest of y's goal that [a world state] z be true at the same time."

Parsons et al. [13] studied conflict between BDI agents and ascertained that conflict happens when:

1. Agents adopt opposing intentions;
2. An agent wants another agent to change a particular mental attitude;

Lastly, we have a definition that divides conflict detection in two stages[14]: potential conflict and true conflict. Consider an agent ag_i operating in an environment populated with other agents that interact with ag_i. Each agent has a set of possible plans about other agents present in the environment and these plans are composed by intentions. Let ag_j be one of the agents present in the environment. The agent ag_i declares a potential conflict if an intention adopted by itself is incompatible with an intention ag_i believes ag_j has. Only after ag_i validates the belief about ag_j's (incompatible) intention, does it declare a true conflict. This approach was adopted in this dissertation.

2.3 Negotiation in Multiagent Systems

Negotiation is a form of social interaction that allows for the resolution of problems occurring in a number of situations. Typically, these situations involve:

- Two or more parties;
- A conflict between the parties;
- An individual preference to find an agreement.

When developing negotiating agents, there are various approaches or models that can be adopted, each having specific strengths and weaknesses. Three well-known models are as follows[15]:

- *Game-theoretic models*: involves the analysis of specific negotiation situations and indicates precise results concerning the optimal strategies negotiators should choose;
- *Heuristic models*: indicates negotiation guidelines and beneficial strategies that lead to good outcomes (and not optimal outcomes);
- *Argumentation-based models*: negotiators can argue about their mental attitudes during negotiation.

Negotiation itself is composed by different processes, notably[6]:

1. *A prenegotiation process*: deals with the preparation and planning for negotiation;
2. *An actual negotiation process*: where the search for an agreement is held;
3. *An execution process*, where a final agreement is analyzed and implemented;

The prenegotiation process is the most important for the present dissertation and a brief description of it follows

Prenegotiaton involves mainly the conception of a plan which delineates the activities that negotiators should attend to before starting the actual negotiation process. The collection of activities to be fulfilled includes [15]:

- Defining and prioritizing the issues;

- Establishing the negotiation agenda;
- Defining the limits, i.e. the points where any settlement beyond them is not acceptable, thus terminating the interaction;
- Defining the targets, i.e. the values negotiators realistically feel they should get in a settlement;
- Selecting an appropriate interaction protocol. The protocol may simply allow for the exchange of offers, or alternatively, support argumentation.

There are three groups of strategies commonly used by (human) negotiators [16]:

- Contending – negotiators maintain their aspirations and try to persuade the opponent to yield;
- Concession making – negotiators cut back their aspirations to accommodate the opponent; these strategies try to find acceptable compromises [17];
- Problem solving – negotiators maintain their aspirations and try to reconcile them with the opponent's aspirations; they work towards integrative agreements, which provides higher joint benefit than compromise agreements [18].

Strategies are typically implemented through a variety of tactics. The main difference between strategies and tactics is the scope. Tactics present short-term actions to enact high-level strategies [19].

The actual negotiation process involves mainly an exchange of offers and counter-offers [20]. However, negotiation involves more than a series of proposed agreements. The exchange of information is related to two dilemmas that all negotiators face [21]. The first is honesty: how much truth to tell the other negotiating parties. In case of too much truth, the opposing agents may take advantage of that information. In case of too little, negotiation may end in a stalemate. The second dilemma is trust: the credence put on the information received by the other partie. This decision takes into account a number of factors, such as how negotiators have negotiated in the past, their negotiation reputation, and the present circumstances.

2.4 Electricity Markets

Electricity is most often produced at power stations, transmitted at high-voltages to multiple substations, and distributed at medium and low-voltages to end consumers. Almost since inception, electric utilities have been vertically integrated, meaning specific firms were accountable for generation, transmission and distribution [22]. This monopoly model has been discarded worldwide over the years, through the deregulation of electricity markets.

Liberalization aims to increase competition, resulting in better pricing and service quality, which in turn leads to more satisfied end-users. To achieve this, deregulation has separated the functions of electricity generation and retail from the natural monopoly functions of transmission and distribution. This has led to the establishment of a wholesale market for electricity generation and a retail market for electricity retailing, where end-users can choose their supplier from competing electricity retailers [15].

2.4.1 Pool Markets

Rather than relying on iterative interactions between suppliers and consumers to reach a market equilibrium, pools provide this equilibrium in a systematic way. A pool generally operates as follows: generating companies submit bids to sell a specific amount of electricity at a certain price and for a certain period, while consumers submit offers specifying the quantity and price at which they intend to buy the energy. Supply and demand curves are generated and intersected. The intersection represents the market equilibrium. All the bids submitted at a price lower than or equal to the market clearing price (the price of the electricity at which the quantity supplied is equal to the quantity demanded) are accepted, and generators are instructed to produce the amount of energy corresponding to their accepted bids. Similarly, all the offers submitted at a price greater than or equal to the market clearing price are accepted, and the consumers are informed of the amount of energy that they are allowed to draw from the system. The market clearing price represents the price of one additional megawatt-hour of energy and is therefore called System Marginal Price (SMP). Generators pay this SMP for every megawatt-hour that they produce, whereas consumers pay the SMP for every megawatt-hour that they consume, irrespective of the bids and offers that they have submitted [23].

2.4.2 Bilateral Contracts

In bilateral contracts, negotiation is performed directly between a party representing the demand and a party representing the offer. The negotiation features a discussion to define prices, volumes, terms and conditions, which after agreed on, will be denoted on a contract [24]. This model allows for a better price stability, increasing the chance to avoid price volatility. There are several types of bilateral contracts, notably [23]:

- *Customized long-term contracts* – the terms of these contracts are flexible since they are negotiated privately to meet the needs and objectives of both parties. They usually involve the sale of large amounts of power (hundreds or thousands of megawatts) over long periods of time (months or even years). The large transaction costs associated with the negotiation of such contracts make them worthwhile only when the parties want to buy or sell large amounts of energy;

- *Trading over the counter* – these transactions involve smaller amounts of energy to be delivered according to a standard profile, that is, a standardized definition of how much energy should be delivered during different periods of the day. This form of trading has much lower transaction costs and is used by producers and consumers to refine their position as delivery time approaches;

- *Electronic trading* – participants can submit the offers to buy energy, or bids to sell energy, directly in a computerized marketplace. All market participants can observe the bids and offers submitted, but do not know the identity of the party that has submitted them. When a party submits a new bid, the software checks to see if there is a matching offer for the bid's period of delivery. If it finds an offer whose price is greater or equal to the price of the bid, a deal is automatically struck and the price and quantity are displayed to all participants. If no match is found, the new bid is added to the list of outstanding bids and remains there until either a matching offer is made, the bid is withdrawn, or it lapses because the market closes for that period. Each time a new offer is entered in the system, a similar procedure is used. This form of trading is extremely fast and cheap.

2.5 Computational Platforms

There are numerous platforms to assist the development of a MAS. This section describes three well-known platforms.

2.5.1 Open Agent Architecture (OAA)

The Open Agent Architecture was developed at the Stanford Research Institute and allows software services to be provided through the cooperative efforts of autonomous agents. Communication and cooperation between agents is brokered by facilitators, which are responsible for matching requests from users (or agents). This process does not require that an agent or a user know the identities, locations, or number of the other agents involved. OAA is structured to minimize efforts in creating new agents and dealing with legacy applications. Figure 2.3 presents the small OAA system, showing a user-interface agent, some application agents and several meta-agents. The agents play different and important roles. All of them are organized in some sort of "community", by being linked to the same facilitator. More than one facilitator is possible, especially in larger systems, creating multiple "communities", each similar to the one shown in figure 2.3.

In some systems, the user interface agent is implemented as a collection of "micro-agents" each monitoring a different input modality (shown in figure 2.3 as "Modality Agents"). Also, every agent includes a set of declarations of the services it provides, in an Interagent Communication Language (ICL). These public declarations are referred to as *solvables* and are composed of three parts [25]: a goal, a list of permissions, and a list of parameters.

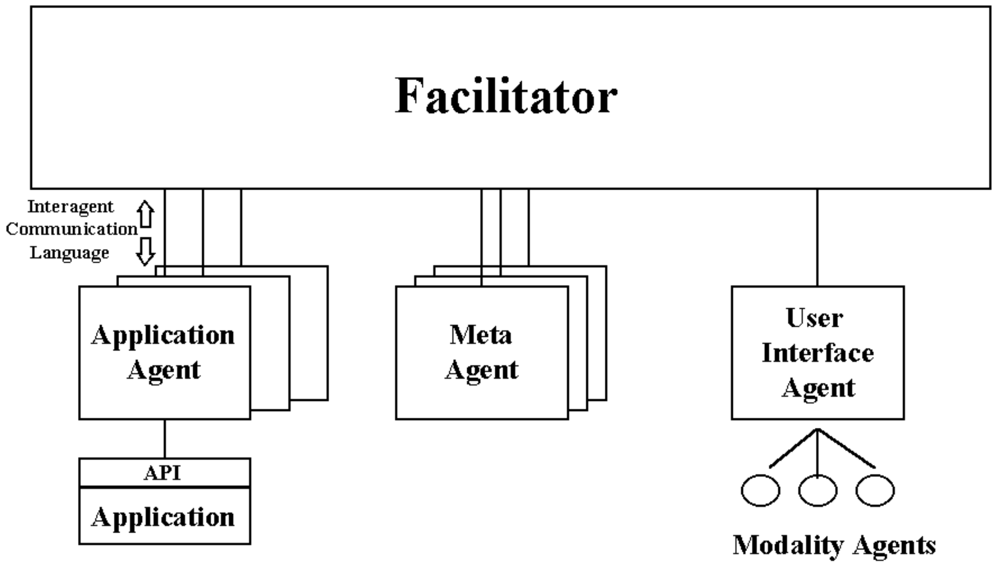


Figure 2.3: OAA System Structure [25]

2.5.2 Repast

Repast was created at the University of Chicago in close collaboration with the Argonne National Laboratory. It is a free and open-source agent-based modeling and simulation toolkit with three released platforms: Repast for Java, Repast for the Microsoft .NET framework, and Repast for Python Scripting. Repast Simphony (Repast S) extends the Repast portfolio by offering a new approach to the simulation, development and execution, including a set of advanced computing technologies for applications (such as social simulation). Although not completely, Repast focuses on social behavior, having been applied to a wide variety of applications, ranging from social systems to evolutionary systems, market modeling, and industrial analysis [26].

2.5.3 JADE

The JADE platform is a middleware technology for the development and run-time execution of peer-to-peer agent based applications, which can interoperate both in wired and wireless environments. JADE is compliant with the Foundation for Intelligent Physical Agents (FIPA) specifications, and provides a homogeneous set of Application programming interfaces (APIs) that are independent from the underlying network and Java version[27]. JADE is probably the most widespread agent-oriented middleware today. It is completely distributed with a flexible infrastructure which allows for easy integration of extensions and ad-on modules. JADE is written in Java and provides a run-time environment implementing the life-cycle support features required by agents [28]. Accordingly, JADE was the platform adopted in this dissertation.

2.6 Energy Market Simulators

Presently, there are several simulators applied to the liberalized market energy. This section describes the following six: EMCAS, MASCEM, SEPIA, AMES and PowerWeb.

EMCAS

The Electricity Market Complex Adaptive System (EMCAS), developed by the Center for Energy, Environmental and Economic Systems Analysis at the Argonne National Lab, is one of the most popular energy market simulators [29]. Agents in EMCAS include the Independent System Operator (ISO), the Generation Company, the Customer, the Demand Company, the Distribution Company, the Transmission Company, and the Regulator. They show decentralized decision-making capabilities, and are able to learn through genetic algorithms [24]. Each agent supports a diverse number of strategies, while simultaneously allowing specific market rule addition by users, granting a posterior analysis of the impact on both individual agents and the global system [29].

EMCAS supports negotiation through bilateral contracts and pool markets. Agents can represent producers, intermediaries, consumers and independent system operators. Bilateral contracts are initiated by the agent representing the demand, followed by a response from the agent representing a producer. The latter is based on foretold prices for pool and bilateral contracts, ascertained through history analysis and future predictions. The decision model of agents is based on an assemblage of public information, made available to all market participants, and private information, resulting from previous proposal acceptance/rejection accounts. Agents can use this information to rethink their strategies for the next round of negotiation [24]. EMCAS is currently utilised by REN for analysis on MIBEL [30].

SEPIA

The Simulator for the Electric Power Industry Agents (SEPIA) was developed by Honeywell Technology Center and the university of Minnesota [31]. It is a specific Agent-Based Simulation tool for developing Energy Market Agent-Based Simulation models with the purpose of gaining insights about the behavior of system participants and their impact on EMs. The main physical components are: Zones, Physical Generators, Generation Companies, Generator of Last Resort, Consumer Load, Consumer Companies, Transmission System, and Transmission Operator [29]. The simulator can be split up into three main components:

1. Graphic interface, which allow users to specify, monitor and drive the simulation;
2. Agents that represent specific domain entities;
3. Discret event simulation mechanisms that control the evolution of the simulation and agent communication.

Agents in SEPIA are composed by diverse layers. Also, agents that represent producers companies have an adaptive module, with a learning component based on the *Q-learning* algorithm, which allows them to explore alternative strategies for pricing. It should also be mentioned that this simulator is limited to bilateral contract simulations [24].

PowerWeb

Developed by the Cornell University, this simulator allows testing and analysis of different kinds of energy markets, and implementation of diverse market models. However, it only supports fixed demand for pool markets. Sellers can bid for energy, from one single production unit allowed to them, in two different ways:

- Explicitly declaring the price and quantity of the bid until the unit production maximum is reached;
- Choosing one of the six available strategies.

This simulator can be accessed through a web application, which allows for usage in distributed computing environment [32]. Agent usage is simple and limited, being its purpose to simply model different types of proposals for market trading.

AMES

The Agent Based Modelling of Electricity Systems (AMES) is an open source multi-agent simulator developed within the JAVA language, for the Repast platform, by *Tsun and Tesfatsion* [33]. With no commercial end in mind, it was developed with the purpose of betterment EM's research and education. This system includes four main components: traders, transmission grids, markets, and an ISO. The market involves a day-ahead market and a real-time market, while the ISO has four functions: system reliability assessment, day-ahead unit commitment, dispatch, and settlement. A learning module is integrated into the simulation framework for adaptive decision making of traders, and the physical transmission grid is modeled as a five-node transmission grid [29].

MASCEM

Developed by the *Instituto Superior de Engenharia do Porto* using the JAVA programming language, the Multi-Agent System that Simulates Competitive Electricity Markets (MASCEM) is a Energy Market Multi-agent System [34]. The agents can simulate market facilitators, generators, consumers, market operators, traders, and network operators. MASCEM features bilateral contracts and a pool market, although on the latter consumers can directly submit their bids to the market operator. Accepted agreements can result from both bilateral contracts and the pool market, and must be submitted for verification by the market operator [29]. During the negotiation process, this simulator allows agents to re-think their strategies, based on results from previous rounds of negotiation [35].

Discussion

The aforementioned complex systems focus mainly on the negotiation process. Important features that help depict real-life negotiations can be found in most of them, such as MASCEM's mid-negotiation strategy reconsideration. A short comparison of these features is illustrated in 2.1. The main agent types portrayed are generators, consumers, and market operators. However, the prenegotiation process is basically not featured in these systems, although individual steps of this process, such as the definition of limits for the negotiation issues, are included (as part of the negotiation process). As stated above, this dissertation focuses on the main prenegotiation tasks, and the link between the individual and social behavior of agents, i.e. the role conflict plays in driving negotiation.

Table 2.1: Comparison between existing Electricity Market (EM) simulators

	EMCAS	**SEPIA**	**PowerWeb**	**AMES**	**MASCEM**
Market models	Bilateral Contracts; Pool	Bilateral Contracts	Pool	Pool	Bilateral Contracts; Pool
Agents	ISO; Generation Company; Customer; Demand Company; Distribution Company; Transmission Company; Regulator	Generation Companies, Generator of Last Resort, Consumer Companies, Transmission System, and Transmission Operator	Sellers; ISO	Trader, market, transmission grid, ISO	Generator, consumer, trader, market facilitator, market operator, network operator
Multi-Agent System	Yes	Yes	No	Yes	Yes
Adaptation	Genetic algorithms	Genetic algorithms; Q-learning for producer agent	No	Variant Roth-Erev (VRE)	Strategy based on history
License	Commercial	Commercial	Free	Open source	Research project

3
Autonomous Agents, Conflict, and Bilateral Negotiation

Contents

3.1 Introduction

A quick search for a definition of “software” leads to [36]:

> “... the instructions which control what a computer does...”

While this still applies to today’s agents architectures, the notion of *autonomous* agent tries to break free from the domain defined in the previous software definition. A look at a definition of the term “autonomous” leads to [36]:

> “... independent and having the power to make your own decisions...”

From the previous statements, it is possible to infer a goal for autonomous agents: to *enhance* original instructions. Also, depending on the application, adaptability and learning may be important features of autonomous agents, allowing computer software to respond to situations not defined in the original instructions, and build on previous situations towards better decision-making algorithms.

This chapter starts by detailing the architecture of BDI agents (as it was adopted by the developed simulator in the dissertation Negotiation Simulator for the Energy Market (NSEM)). Next, conflict and bilateral negotiation are discussed, placing emphasis on the prenegotiation phase of negotiation, and detailing the negotiation strategy developed for this work. Finally, the chapter introduces a case study involving a retailer and a customer negotiating bilateral contracts in an Electricity Market (EM). The case study illustrates some aspects of the individual and social behavior of agents, placing emphasis on plan generation and issue identification.

3.2 Autonomous Agents

The Belief-Desire-Intention architecture tries to model the human decision-making behavior and has its roots on the practical reasoning process [10]. Its main elements are:

- A set of beliefs, representing information about the agent itself, the other agents, and the environment;
- A set of goals, representing world states to be achieved;
- A library of plan templates, representing actions that should be performed to achieve goals;
- A plan generator, responsible for obtaining the required information from the previous three elements, conjugate them, and create full plans;
- A set of adopted plans for execution, either immediately or in the near future.

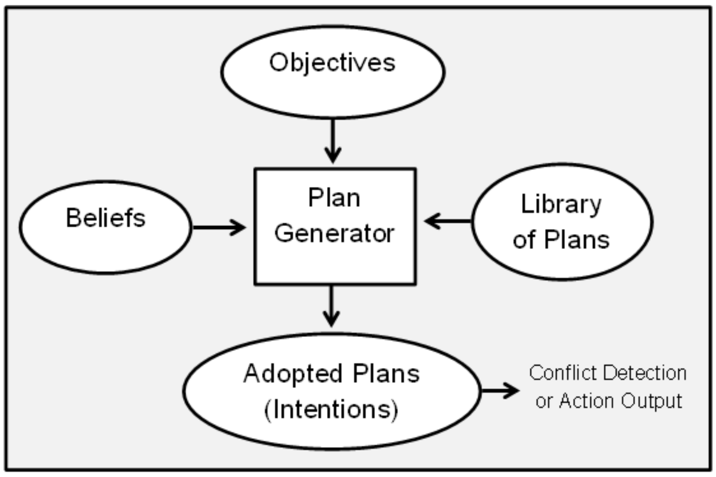

Objectives
Beliefs
Plan Generator
Library of Plans
Adopted Plans (Intentions)
Conflict Detection or Action Output

Figure 3.1: The Belief-Desire-Intention architecture

Figure 3.1 shows the main components of the architecture, and how they interact.
A description of each component follows (see Lopes et al. [14] for a more in depth description).

Let $Ag = \{ag_1, ag_2\}$ be a set of autonomous agents. Let $ag_i \in Ag$ be an agent from Ag and $B_i = \{b_{i1}, b_{i2}, ...\}$ a set of beliefs of ag_i. Each belief $b_{im} \in B_i$ is represented as follows:

$$b_{im} = < bh_{im}, bb_{im} >$$

where:

- **(i)** bh_{im} is the header of b_{im};
- **(ii)** bb_{im} is the body of b_{im}, and contains a statement that ag_i believes to be true.

The header bh_{im} of b_{im} is a 2-tuple: $bh_{im} =< name, bd_{im} >$, where $name$ is the name of an agent in Ag of which the intention is about, and $descr_{im}$ is a short description of b_{im}.

Let $G_i = \{g_{i1}, g_{i2}, ...\}$ be the set of goals that ag_i wants to achieve. A generic goal $g_{ik} \in G_i$ is described as follows:

$$g_{ik} =< gb_{ik} >$$

where gb_{ik} is a statement that identifies the goal g_{ik}. This statement is used by the "Plan Generator" to search for an adequate plan template from the plan library. The library consists of a collection of plan templates, $PL_i = \{pt_{i1}, pt_{i2}, ...\}$, where plan template $pt_{iz} \in PL_i$ is a 5-tuple [14]:

$$pt_{iz} =< pth_{iz}, ptt_{iz}, ptpc_{iz}, ptb_{iz}, pte_{iz} >$$

where:

(i) pth_{iz} is the header of pt_{iz}; it is the attribute used when searching for a plan template, which is compared to a goal's body;

(ii) ptt_{iz} is the type of pt_{iz}, either composite or primitive;

(iii) $ptpc_{iz}$ is a set of preconditions that must be true before pt_{iz} can be applied;

(iv) ptb_{iz} is the body of pt_{iz}; its content depends on the type;

(v) pte_{iz} is a set of statements that should hold after pt_{iz} has been successfully executed.

The header pth_{iz} is itself a 2-tuple, i.e. $pth_{iz} = <ptn_{iz}, pta_{iz}>$, where ptn_{iz} is the name of pt_{iz} and pta_{iz} is a set of arguments for pt_{iz}.

There are two possible types of plan templates, namely primitive and composite. If pt_{iz} is a *primitive* plan template, then ptb_{iz} is a set of one or more actions that satisfy a goal. On the other hand, if pt_{iz} is a *composite* plan template, ptb_{iz} is a collection of references to other plan templates present in the library. In other words, a composite plan template specifies the decomposition of a goal into sub-goals, by referencing other composite or primitive plan templates.

An agent $ag_i \in Ag$ is capable of generating complex plans, $\{p_{i1}, p_{i2}, ...\}$, based on the simple plan templates stored in the library. The process of generating a plan p_{iy} for achieving a goal g_{iy} involves four main tasks [15]:

- Plan retrieval. Upon choosing g_{iy}, the agent ag_i uses it to search the library PL_i for all plan templates whose header unifies with the body of g_{iy}, and whose preconditions $ptpc_{iz}$ apply in the current state. The plan templates retrieved are labeled applicable plan templates. The set of applicable plans is denoted as APT_{iy};

- Plan selection. Consists of selecting a preferable plan template, let's say $pt_{iz} \in APT_{iy}$. The selection is based on a scoring system to find the most favorable plan template;

- Plan addition. The plan generator adds the previously selected plan template pt_{iz} to the plan p_{iy}. Let $AltP_{iy} = APT_{iy} - pt_{iz}$. The plan templates in $AltP_{iy}$ are added to p_{iy} and labeled as *alternative plans*. They have a key role in prenegotiation (see in sub-section 3.5.2);

- Plan interpretation. This task consists of selecting a composite plan template pt_{iz} from the plan p_{iy} and *expanding* its body's elements, which in composite plans are themselves goals. Accordingly, the plan generator repeats these four tasks for each of those goals, resulting in a recursive process that ends when all primitive plan templates have been retrieved, selected and added to the plan.

Let $IS_i = \{int_{i1}, int_{i2}, ...\}$ be the Intention Structure (IS) of ag_i. The several elements of IS_i are plans adopted by ag_i, i.e. intentions, and are structured into *and-trees*. In particular, after generating the plan p_{iy}, the agent ag_i *adopts* p_{iy} for execution in the future. This is accomplished by recursively adding every plan template present in p_{iy} into the IS_i, as an intention to execute, i.e. a goal that is not yet achieved but is considered, at the present moment, achievable. Each intention created by agent ag_i corresponds to an header of a plan template, thus corresponding to a goal that ag_i wishes to reach.

3.3 Conflict of interests

Much like human negotiators, an agent in a negotiation environment needs two facets:

- A private side, which considers the agent's own rationalization process, its beliefs, individual goals, strategies, intentions, etc. This is also called the individual behavior component;
- A public side, accountable for any interaction with other agents (e.g., negotiation and communication), requiring knowledge and skills on how to deal with social situations. This is also known as the social behavior component.

Although both sides can exist individually, each side has less meaning and purpose without the other. Without interaction, goals might not get achieved and beliefs may stay outdated. Without the individual component, would there be a reason for such interaction? Probably not.

An agent with both components also needs a way to determine how they should interact. This is the key role of conflict. In a MAS, with multiple agents operating individually to meet their own design goals, conflict is inevitable. Let $int_{iq} \in IS_i$ be an intention that $ag_i \in Ag$ wants to accomplish. Likewise, let $int_{jk} \in IS_j$ be an intention that the other agent $ag_j \in Ag$ wants to accomplish. A conflict $Conf_{i,j}$ occurs when the intentions int_{iq} and int_{jk} are incompatible [14]:

$$Conf_{i,j} = \exists int_{iq} \wedge \exists int_{jk} \wedge Incomp(int_{iq}, int_{jk})$$

where $Incomp(int_{iq}, int_{jk})$ represents an incompatibility between intentions int_{iq} and int_{jk}, i.e the two intentions cannot be executed simultaneously.

3.3.1 Potential Conflict

Let $PP_i = \{p_{i1}(ag_j), p_{i2}(ag_j), ...\}$ and $PI_i = \{int_{i1}(ag_j), int_{i2}(ag_j), ...\}$ be a set of possible plans and a set of possible intentions ag_i believes ag_j has generated. Agent ag_i checks frequently the existence of any conflict between its intentions and the intentions it believes ag_j

has. Since ag_i is unsure about the true plans and intentions of ag_j, it will initiate a process called *Potential Conflict Detection*, involving the detection of a potential conflict $PotConf_{i,j}$ [14]:

$$PotConf_{i,j} = \exists int_{iq} \wedge \exists int_{im}(ag_j) \in PI_i \wedge Incomp(int_{iq}, int_{im}(ag_j))$$

where:

- **(i)** $int_{im}(ag_j) \in PI_i$ represents an intention ag_i believes ag_j has;
- **(ii)** $Incomp(int_{iq}, int_{im}(ag_j))$ represents an incompatibility between int_{iq} and $int_{im}(ag_j)$.

If $PotConf_{i,j}$ is true, ag_i needs to *validate* it by confirming the veracity of the intention $int_{im}(ag_j)$. This process involves a conversation between ag_i and ag_j, where ag_i informs ag_j about the potential conflict and requests this agent to validate the intention $int_{im}(ag_j)$. The agent ag_j can choose either to refuse to participate in a conversation, or to reply to the message received. If ag_j confirms $int_{im}(ag_j)$, ag_i will declare the conflict as *validated*, and inform ag_j. If ag_j does not confirm $int_{im}(ag_j)$, ag_i will request information about the true intention of ag_j, so to double-check for the existence of a conflict, and consequently be able to either declare the conflict as *validated*, or to acknowledge the existence of no conflict. In either case, ag_j is informed before ending the conversation.

3.4 Bilateral Negotiation

As stated before, three main phases are often identified in negotiation: prenegotiation, actual negotiation, and execution (or ending). The prenegotiation phase deals with the activities that come before formal negotiation. The actual negotiation process involves mainly the exchange of offers between the parties. It may also feature argumentation, learning, dynamic strategy selection, and impasse resolution. The last phase, execution, includes the analysis and improvement of a final agreement. Since prenegotiation is the main focus of this work, the remainder of this section presents some details of the most importante prenegotiation activities.

3.4.1 Prenegotiation

Is the starting point of negotiation equally fair to all participants? Any confrontation between humans suggests that this is not the case. Personal traits will distinguish individuals in such situations, be their personality, tolerance, creativity, and many other attributes. The same holds for negotiation, where traits such as charisma, confidence, boldness, and so on, give an edge to negotiators. Clearly, such traits are not easily obtained. To some negotiators, they may be inherent. Others may find them with experience. However, there are other methods to obtain the so sought after advantage, even before the starting of negotiation. Accordingly to successful negotiators, preparation and planning, also known as prenegotiation, is central to obtain a successful outcome, and the aforementioned advantageous traits cannot overcome a poor planning.

Prenegotiation is the first step in the overall negotiation process, composed by the following phases [15]:

- **(i)** the definition of the issues to negotiate and the creation of a negotiation agenda;
- **(ii)** the definition of the limits and targets for each issue at stake;
- **(iii)** the selection of an interaction protocol, and the definition of the preferences;
- **(iv)** the selection of a negotiation strategy.

A brief description of each of these tasks follows.

Agenda

The negotiation agenda, i.e. the collection of issues at the negotiation table need to be agreed by all parties. To this end, each agent submits its own proposal about the set of issues to discuss during negotiation. This proposal (or list) is formed from the headers of all the primitive plans associated with the negotiation plan. These primitive plans represent facts that the agent wishes to discuss during negotiation.

The agenda is thus the result of an interaction process. The initiator submits a proposal and the opposing agent sends a counter-proposal. If the initiator agent accepts the counter-proposal, or both the proposal and the counterproposal contain the same issues, an agreement is reached and the agenda is defined.

Targets and Limits

The agents must assign target and limit values for each issue on the agenda. A target is a level of benefit to stride for at any time. A limit is a fallback position, i.e. a level of benefit beyond which an agent will not yield.

Deadline

A mutual deadline must be agreed by both negotiating sides before the starting of negotiation. The deadline is a date and time set for the near future.

The Negotiation Protocol

The negotiating parties must agree on an appropriate negotiation protocol. The protocol adopted in this dissertation is the alternating offers protocol [37]. A brief description of it follows.

Let ag_i and ag_j be two agents, each representing a negotiating party. Either agent may start the negotiation process, by sending the first proposal. A proposal $prop^t_{sender,receiver}$ is a collection of issues with attached values. A counterproposal is structurally a proposal sent in response to a previous proposal. Let ag_i be the first agent to submit a proposal. Let $t = \{0,1,2,...\}$ be the set of time periods. Every proposal from ag_i will correspond to an even value for t. Thus, ag_j will always send its proposals at odd values for t.

The first proposal, $prop^{t_0}_{ij}$, is sent from ag_i to ag_j . Next, ag_j receives $prop^{t_0}_{ij}$ and may either: (i) accept $prop^{t_0}_{ij}$, (ii) reject $prop^{t_0}_{ij}$ and inform about the end of negotiation, or (iii) reject $prop^{t_0}_{ij}$ and send a counterproposal $prop^{t_1}_{ji}$. If ag_i receives from ag_j: (i) a proposal acceptance, then negotiation ends with ag_i informing ag_j of the final agreement, (ii) a proposal rejection, then negotiation ends, (iii) a counterproposal, then ag_i has the same options described previously for ag_j, i.e. it can accept $prop^{t_1}_{ji}$, reject $prop^{1}_{ji}$ and end the conversation, or reject $prop^{t_1}_{ji}$ and send a counterproposal $prop^{t_2}_{ij}$. This process is repeated until either a proposal is accepted by both agents (negotiation ends with an agreement), or a deadline is reached (negotiation ends with no agreement). Figure 3.2 shows a block diagram of this process.

3.4.2 Preferences of Negotiators: The Additive Model

The preferences of the negotiators are important to evaluate both received and sent proposals. The additive model is adopted in this work because of its simplicity.

The additive model in a well known preference model [38]. It considers a utility function and is used in situations where agents negotiate different issues. The utility associated with a value v_x of an issue x is computed as follows:

$$V_{ix} = weight_{ix} \times v_x \tag{3.1}$$

where $weight_{ix}$ is the weight agent ag_i gives to issue x. The utility for an entire proposal is obtained from the summation of the utilities for each issue value:

$$U_i = \sum_{x=1}^{z} V_{ix} \tag{3.2}$$

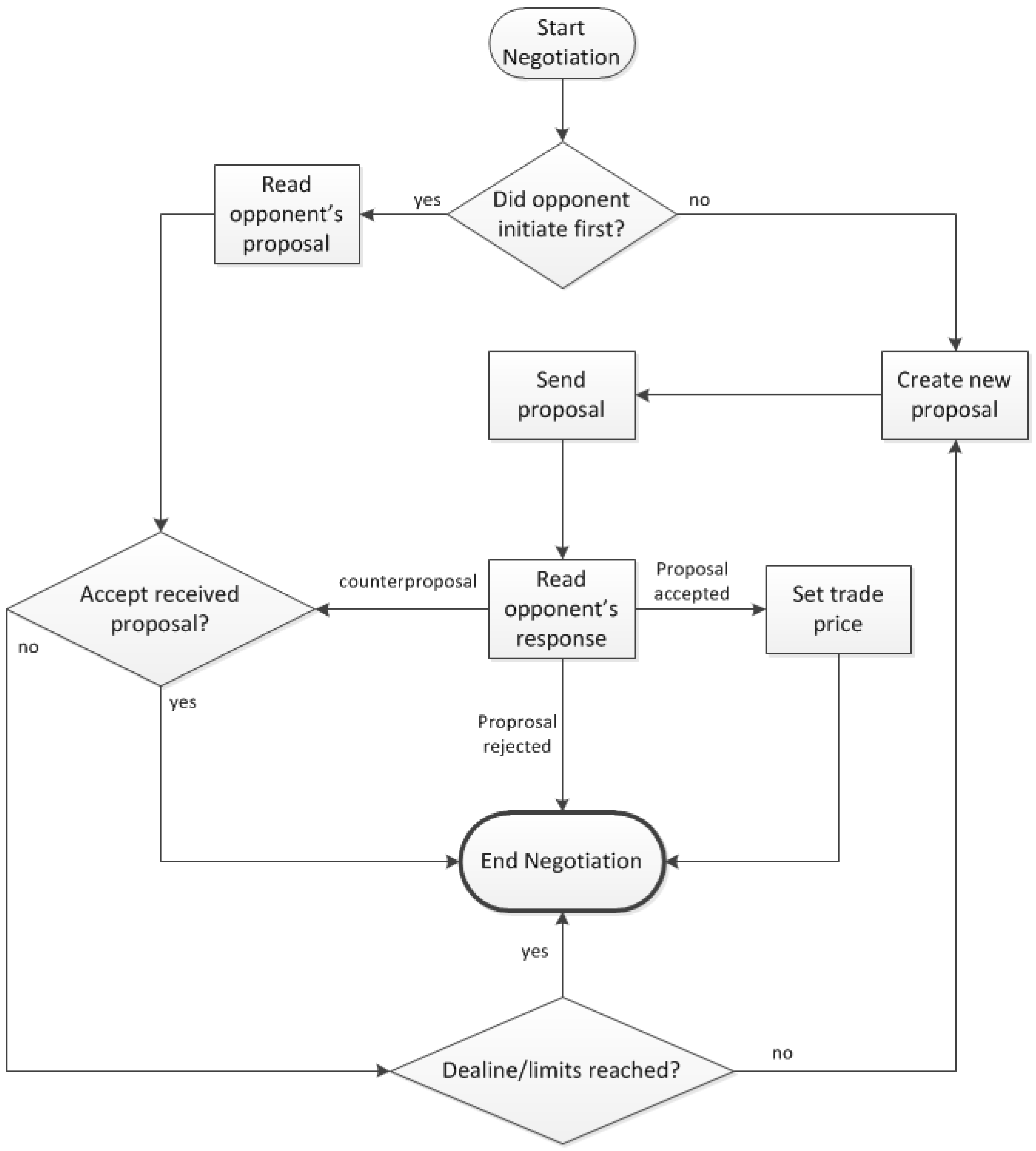


Figure 3.2: Block diagram of the negotiation process

Negotiation Strategies: Random Tit-For-Tat

Strategies are functions that define the tactics to be used at any particular moment of the negotiation process. Tactics define the *actions* made at each point during the negotiation. Strategy choice is an important step, that considers all the available information and has a great impact on the final agreement. There are numerous strategies that can be implemented. For the purposes of this dissertation, the Random Tit-for-Tat strategy was developed [39]. This strategy has proved to be important in cooperative negotiation situations.

The developed strategy is behavior depended, i.e. it computes the next offer based on previous behavior of the opponent. Specifically, an agent reproduces in absolute terms the behavior that the opponent performed in the last step, plus a random value. For instance, let ag_i and ag_j be the two negotiating agents. Let $I = \{x_1, x_2, ...\}$ be the set of issues at stake. If ag_i receives a proposal from ag_j offering x_n with the value v_x one unit less than v_x offered in the previous offer, the buyer will reciprocate by adding one unit to its value for x_n, plus a random value. The strategy can be represented as follows:

$$prop_{ij}^{t+1}[x_n] =$$
$$min\left(max\left(prop_{ij}^{t-1}[x_n] + \left(prop_{ji}^{t-2}[x_n] - prop_{ji}^{t}[x_n]\right) + (-1)^s + R(M), min_{ix_n}\right), max_{ix_n}\right)$$

where:

(i) $s = 0$ if the agent wants to minimize the value of x_n;

(ii) $s = 1$ if the agent wants to maximize the value of x_n;

(iii) $t > 1$;

(iv) min_{ix_n} represents the minimum value of x_n;

(v) max_{ix_n} represents the maximum value of x_n;

(vi) R(M)∈[0,M] is a random value.

The integer M is determined as follows:

$$M = \left[\frac{prop_{ij}^{t}[x_n] - prop_{ij}^{t-2}[x_n]}{2}\right] \times (1 - weight_{ix_n}) \tag{3.3}$$

where $weight_{ix_n}$ is the weight of an issue x_n.

3.5 Case Study

This section introduces a case study involving two agents, a buyer and a seller. The buyer is named John Doe, and the seller NER. The agents wish to trade electricity, and to that end, they generate plans to meet their design objectives. Also, they enter into both a prenegotiation stage and a negotiation stage, in order to reach an agreement. For the sake of readability, this section illustrates the plan generation process and the prenegotiation phase from the point of view of the buyer. Chapter 5 presents an in depth description of the case study

3.5.1 Planning

The buyer agent creates a plan specifying in its root a plan template describing the negotiation goal. Figure 3.3 presents a simplified plan, where each composite plan template, from left to right, is associated with a set of other plan templates, i.e. its subgoals. To achieve a plan template, some or all of its subgoals must be reached (depending on the And-Or associations). This means that in order to achieve the goal described in the plan template located in the plan's root, *Buy_Energy*, the subgoals *Prices* and *Volumes* must also be achieved. The same is true for all subgoals, which in the case of *Prices* means the subgoals $Price_1, Price_2, Price_3, Price_4, Price_5$, and $Price_6$, need to be reached.

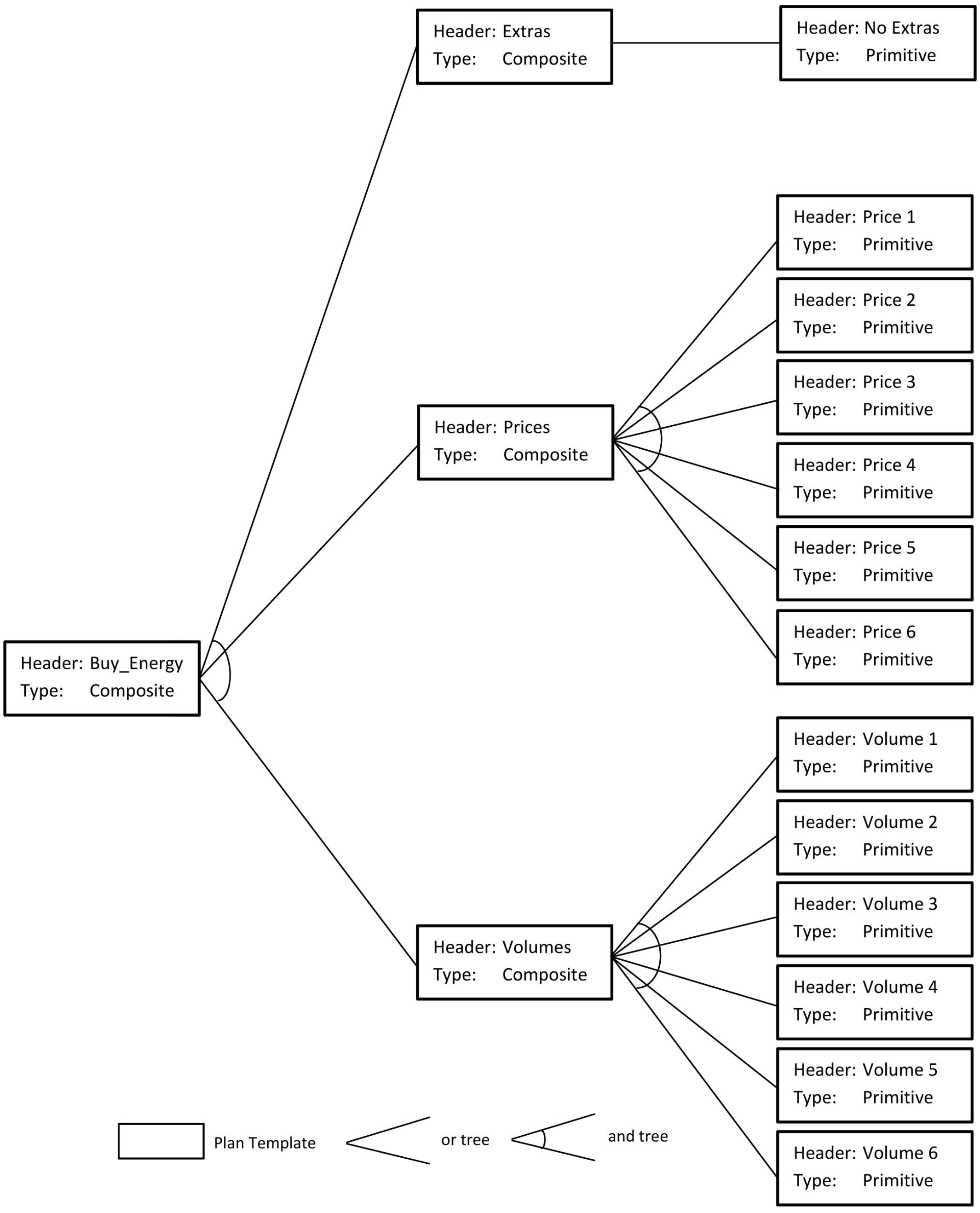


Figure 3.3: A fully composed plan

3.5.2 Conflict and Alternative Plan Expansion

After the plan is inserted into the Intention Structure (IS), the agent checks for the existence of any conflict. If a conflict is found and validated, the original plan should be *expanded* to include all the alternative paths. Specifically, the plan is expanded by inspecting all of its plan templates, in the same order they were added to IS. Each plan template should be checked for alternative plans (stored during plan addition). Figure 3.4 illustrates this process, where plan template pt_{i1} is the original plan and pt_{i2} represents an alternative plan template stored in pt_{i1}.

The addition of a new set of subgoals (pt_{i2}'s body) to the body of pt_{i1}, in a different position, leads to plan template pt_{i3}. Each position on pt_{i3}'s body represents an alternative way to achieve the goal described in pt_{i3}'s header, which means they form an *or-tree*. Figure 3.5 shows part of the plan shown in Figure 3.3, illustrating the expansion of the plan template "Extras". It identifies two alternative sets of sub goals, any of which could be achieved to reach the plan template *Extras*.

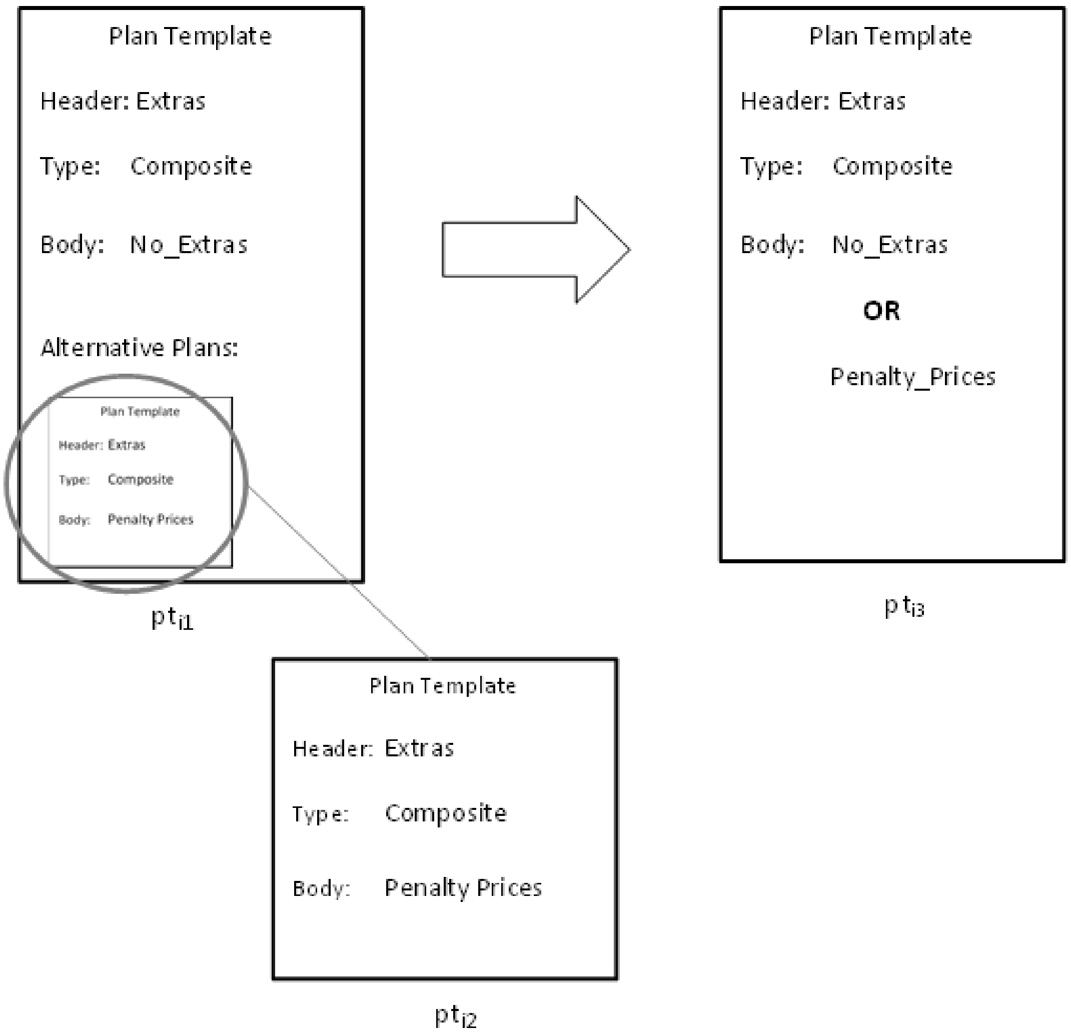


Figure 3.4: Expansion of alternative plan templates

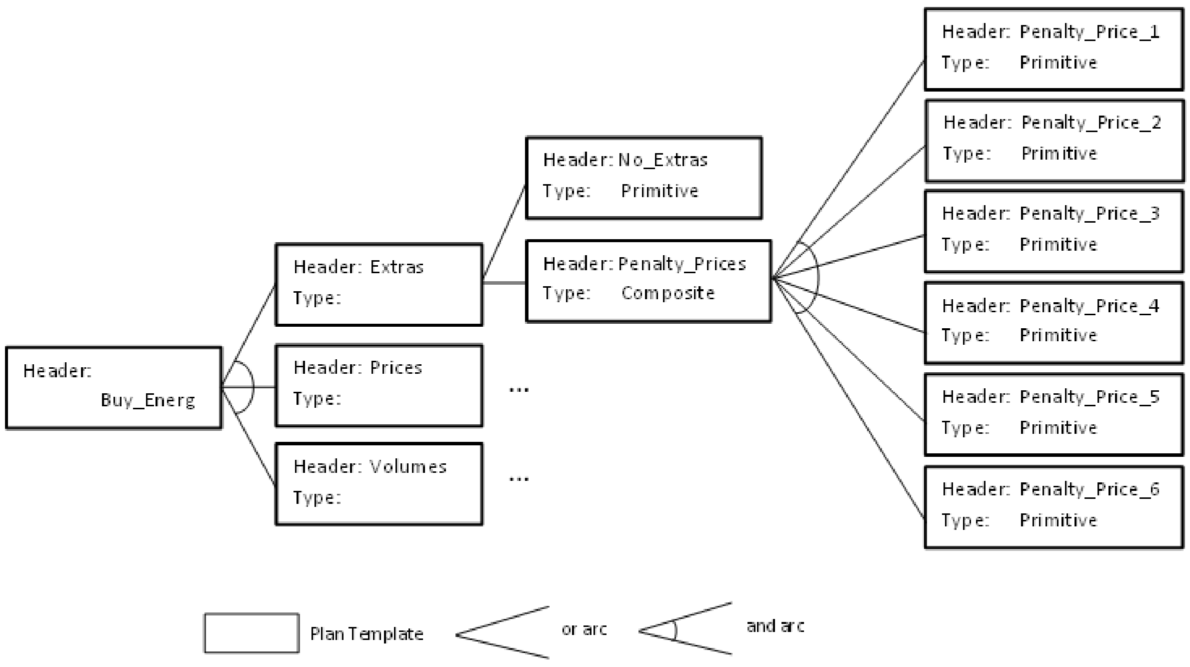


Figure 3.5: A partially composed plan with alternative plans expanded

As stated, the case study will be more detailed in chapter 5. More specifically, chapter 5 will illustrate the definition of the issues to negotiate, the creation of the negotiation agenda, the definition of limit and target values for each issue, the selection of an interaction protocol, the definition of the preferences, and the selection of a negotiation strategy.

4

Negotiation Simulator for the Liberalized Electricity Market

Contents

4.1 Introduction

This chapter presents details of the Negotiation Simulator for the Energy Market (NSEM), i.e. the simulator developed in this dissertation. While the structure of the simulator allows for modifications enabling it to be applied in diverse business environments, it was specified for the negotiation of bilateral contracts in the EM, where six periods of the day were considered. Some concepts were simplified because they fell outside the dissertation objectives. NSEM was developed with the Java language [1] and uses the JADE platform for agent creation, management, and communication.

4.2 System Agents

NSEM includes two main agents: a buyer and a seller. The agents main tasks are to detect, validate and resolve conflicts, the latter being through negotiation. A market agent is also featured to support the negotiation process. A description of each agent follows:

Market Agent

There is only one market agent per system session. Its purpose is to manage newly connected agents and oversight negotiation. Upon connection to the environment, a seller/buyer agent informs the market agent of its existence and indicates both personal information and intentions regarding energy trading. Through the market agent, the user introduces sellers to buyers.

Buyer Agent

The buyer is the more pro-active agent. It is this agent that looks for potential conflict, motivating a social interaction with the seller. The internal process and different possible paths within the buyer's execution flow are represented in figure 4.1.

When a buyer is created, it contacts the market agent in order to find an agent interested in selling energy. Upon being pointed to a specific seller, the buyer loads its objectives, plan template library, and persistent beliefs, i.e. statements the agent believes to be true about itself, the opposing agent and the environment. [2] These beliefs are stored in external files, in order to feature persistent sets of beliefs between different system sessions. Saving the current set of mid-session beliefs to an external file is also possible, allowing the agent to save its current beliefs at any time.

[1]Excerpts from NSEM's Java source code are presented in sections A.1, A.2, and A.3.

[2]Objectives, plan template library, and beliefs are stored and represented in Extensible Markup Language (XML). Some examples of plan templates are presented in A.4.

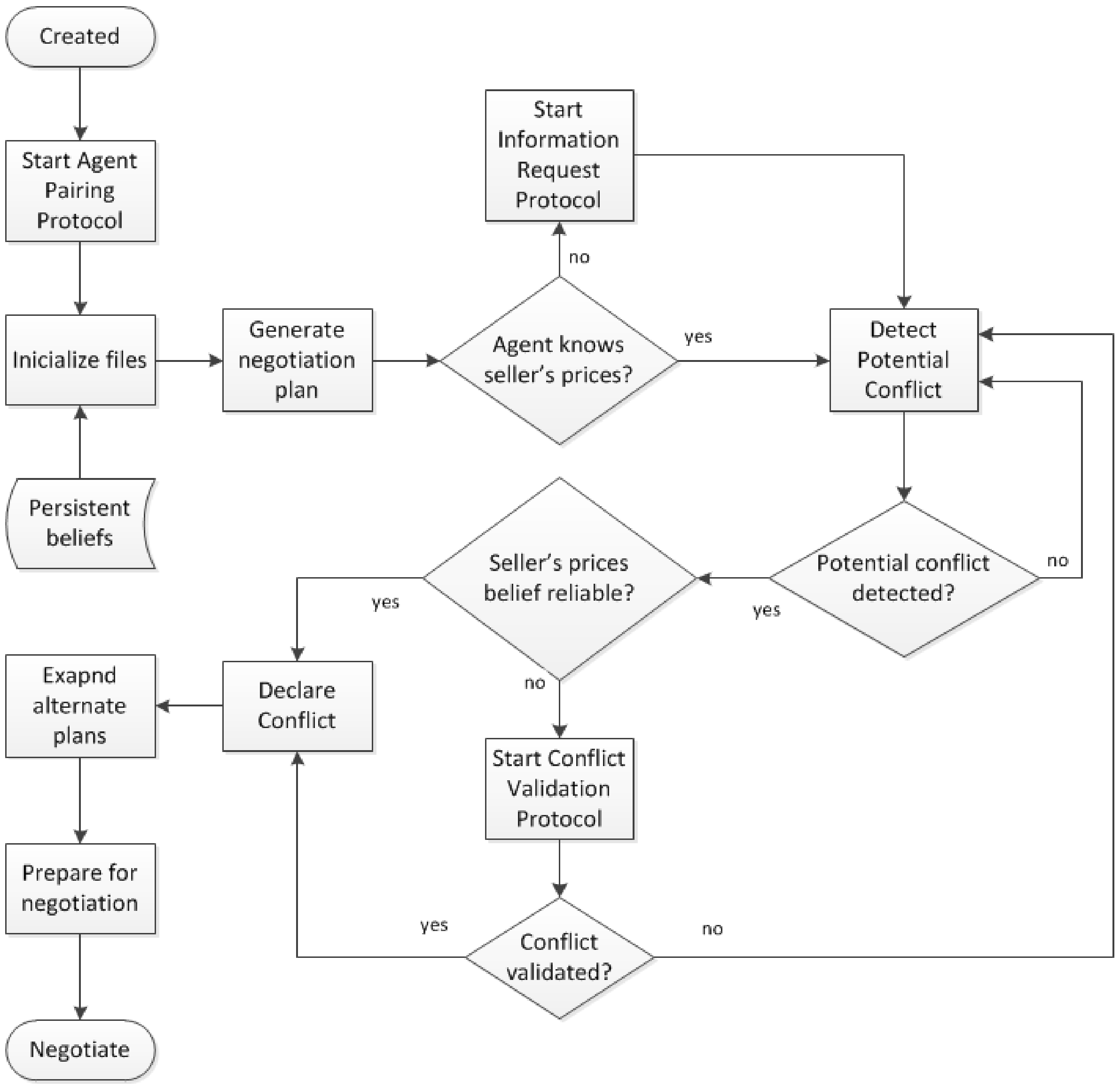

Created
Start Agent Pairing Protocol
Inicialize files
Persistent beliefs
Generate negotiation plan
Agent knows seller's prices?
no
Start Information Request Protocol
yes
Detect Potential Conflict
Potential conflict detected?
no
yes
Seller's prices belief reliable?
yes
no
Declare Conflict
Exapnd alternate plans
Prepare for negotiation
Negotiate
Start Conflict Validation Protocol
Conflict validated?
yes
no

Figure 4.1: Internal flow of the buyer agent

The agent then reads its next goal (the first item on the list of objectives), and search the plan template library for a plan template with the same name as the goal, which will be the negotiation goal. After creating a full plan, the agent tries to detect a potential conflict (between its plan and the opponent's plan). To this end, the buyer needs to know the seller's asking prices, and thus searches for a belief containing this information. If the buyer does not possess such information, it starts the *Belief Request Protocol* in order to request it from the seller.

Following the obtention of the required information, either directly from the source, from previously received publicity, hearsay, or even by user-input (by manually adding the belief to the agent's belief file), the buyer checks for a conflict. If a conflict is detected, the buyer will either:

(i) declare a potential conflict, inform the seller of the potential conflict, and start the *Conflict Validation Protocol*, in case the seller's prices used to check for conflict are not considered *reliable*;

(ii) declare a conflict and inform the seller of it, in case the seller's prices used to check for conflict are *reliable*.

A seller's price belief is considered *reliable* if the source of the information is the seller agent itself, and the information was communicated recently, i.e. during the current system session. For instance, if the seller informed a potential buyer of its selling prices, there is no need for the buyer to validate those prices (and in turn the associated conflict), during the current system session, thus changing the *potential conflict* declaration to a true *conflict* declaration.

If no conflict is declared, then the agent will continue to look for the existence of a conflict (using new price beliefs, if any received). However, if a true conflict is declared, then the negotiation plan should be expanded to include alternative ways to accomplish the goals.

NSEM allows for alternative ways to meet the negotiation goal, for instance, by considering penalty prices (prices that are applicable if the amounts of electricity contracted are surpassed). Accordingly, an alternative negotiation path including penalty prices for the buyer is defined when a conflict is detected (see subsection 3.5.2).

At this stage, it is important to mention that the buyer can send a message to the opposing agent relaying its profile and the associated volumes, i.e. the set of volumes that are in tune with its negotiation intentions.

Seller

The seller agent is similar to the buyer featurewise and its details are thus omitted. For simplicity reasons, this agent does not perform conflict detection. More specifically, this work assumes that the initial prices defined by the buyer agent are generic, i.e. do not refer to any particular seller. This procedure intends to simulate the well-known procedure of indicating the prices of specific commodities (e.g., a car, a house, etc.) before starting the negotiation. Figure 4.2 shows the internal flow of the seller.

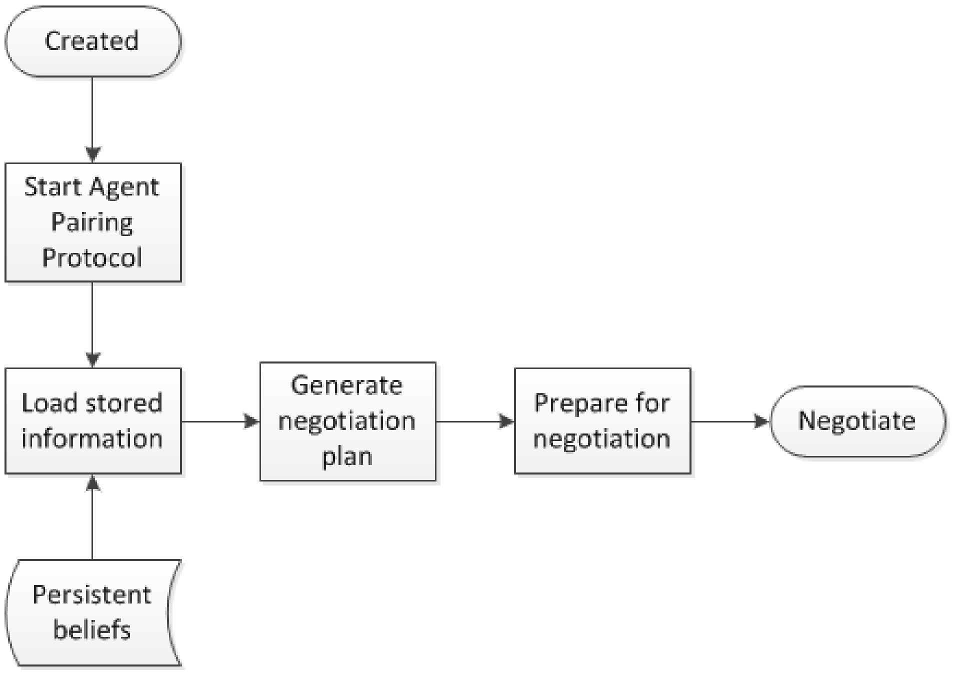


Figure 4.2: Internal flow of the seller agent

4.3 Communication Protocols

JADE provides a diverse number of services and libraries, facilitating the process of agent creation, management and interaction. This platform is compliant with FIPA specifications, providing asynchronous Agent Communication Language (ACL) message passing and agent life-cycle management. A FIPA ACL message is composed of one or several parameters. The number of parameters varies from situation to situation. The only mandatory parameter is the *performative*, but other important parameters are generally used, such as the *sender*, the *receiver* and the *content*. User-defined parameters are also permitted, although their semantics are not defined by FIPA [40].

The agents feature various communication protocols, designed to meet the particular needs of the application. They assure the quality of the information sent and received, and are crucial in managing social tasks. Protocol setup takes advantage of the features provided by the JADE platform, allowing for multiple protocols running at the same time. All protocols start in the same way, namely a message from one agent requesting the opposing agent to take part in a specific protocol. The opposing agent can refuse, ending the interaction, or agree, accepting to take part in a conversation. To end a protocol, one of the agents sends a message informing the opponent of protocol termination.

This section describes the several agent protocols using FIPA notation [41].

4.3.1 Agent Pairing Protocol

This protocol is executed every time a seller or a buyer agent is created, in order to connect it to the market agent, and in turn to pair it up with an opposing agent. Let the agents be identified by ag_i and ag_j. After initiating the protocol, ag_i introduces itself to the market agent by sending a message containing personal information. This is followed by a second message, namely a request for defining a negotiating partner, with compatible trading intentions. When the market agent decides to pair ag_i with ag_j, ag_i receives a message indicating ag_j's name (the same is true for ag_j, but in a different protocol instance). After this, ag_i terminates the protocol.

4.3.2 Belief Request Protocol

This protocol can be initiated by both agents. Its purpose is to ask for specific information. Figure 4.3 shows the FIPA representation of the protocol. After initiating the protocol, ag_i sends a request to ag_j, specifying an agent name and a belief description. The agent ag_j can either search for a belief (using the received belief description) and send it to ag_i, or refuse to participate in the interaction. After receiving the response, ag_i ends the protocol.

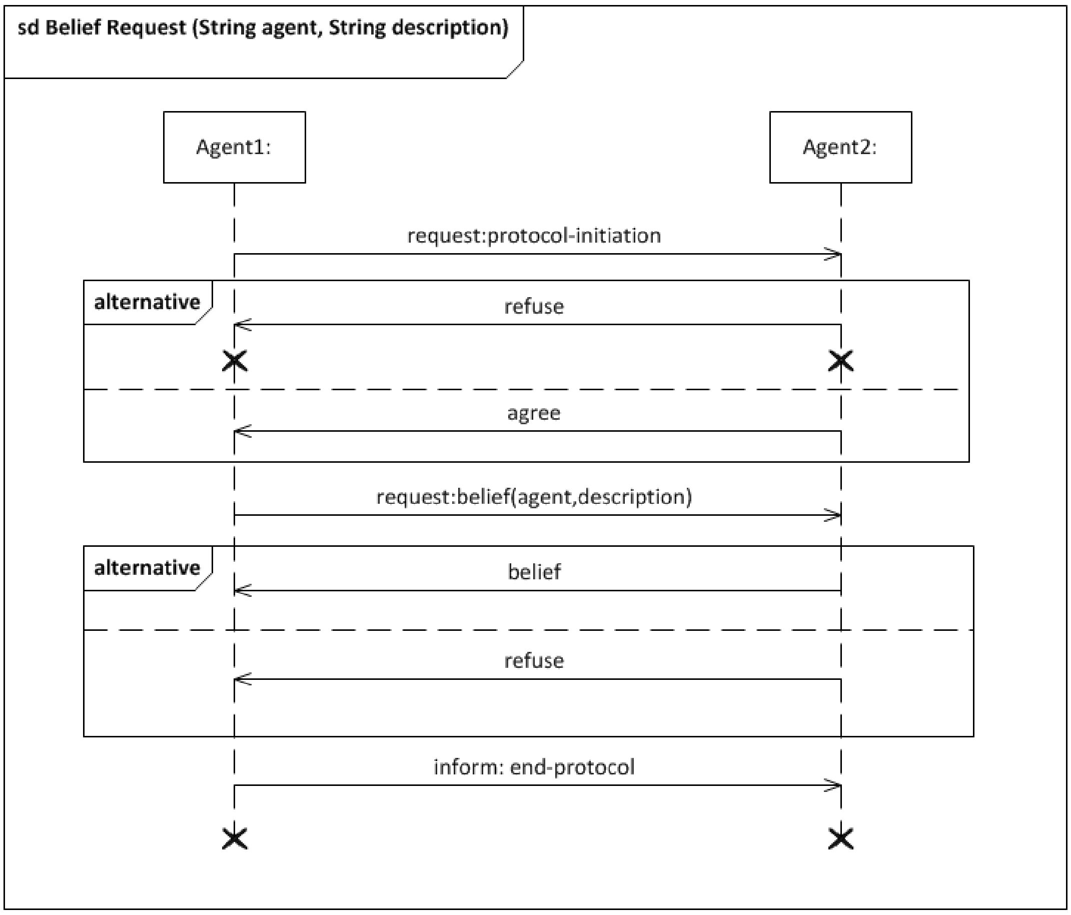

sd Belief Request (String agent, String description)
Agent1:
Agent2:
request:protocol-initiation
alternative
refuse
agree
request:belief(agent,description)
alternative
belief
refuse
inform: end-protocol

Figure 4.3: Belief request protocol

In NSEM, this protocol is used by the buyer to detect a potential conflict. If the buyer does not possess information about the seller's intended prices, it will automatically initiate this protocol, with the following values: Belief Request (*name*, prices), where *name* represents the seller's name.

4.3.3 Agenda Definition Protocol

This protocol is initiated by the buyer agent and aims to cooperatively define the negotiation issues. After initiating the protocol, the buyer generates and sends a proposal that includes all the issues it intends to negotiate. The seller receives the proposal and makes a counterproposal, by analyzing the received proposal and choosing either to delete some issues, to accept all the issues, or to add new issues. Next, the buyer analyzes the counterproposal and decides either to accept or reject it. In either case, the buyer sends a message ending the protocol.

4.3.4 Conflict Validation Protocol

In NSEM, this protocol is always initiated by the *buyer* agent, and its purpose is to allow both agents, the *buyer* and the *seller*, to validate a potential conflict between them (see figure 4.4). After detecting a potential conflict (in particular a conflict between the prices of the buyer and the prices it believes the seller intends to sell electricity for), the buyer will contact the seller to initiate the conflict validation protocol. If the seller agrees to take part in the interaction, the buyer will ask it to confirm a belief about energy prices, i.e. the prices the buyer believes the seller intends to sell for. Two scenarios are now possible:

(i) the seller confirms the prices sent by the buyer; the buyer then informs the seller that a true conflict exists between them, and sends a message informing that the protocol has ended, thus finishing the involvement in the protocol (and upon reception, so does the seller);

(ii) the seller does not confirm the buyer belief.

Upon information that a belief was false or outdated, the buyer requests information about the sellers current prices. The seller then responds by sending its electricity prices. Next, the buyer will double-check for conflict using the new prices, and if the operation returns true, then it sends a message informing the seller of the validated conflict. In case of no conflict was detected, it informs about the existence of no real conflict. The buyer will then end the protocol.

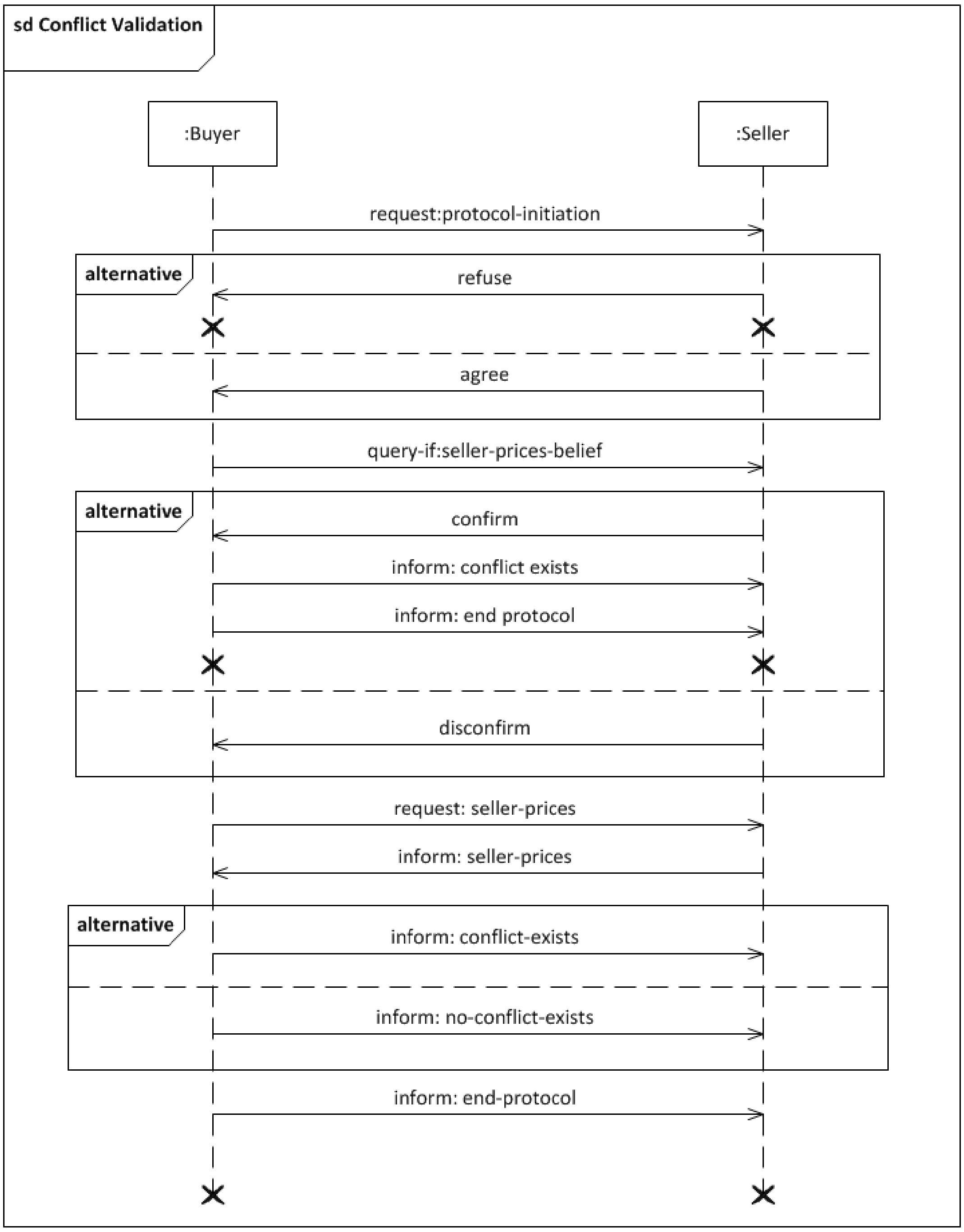


Figure 4.4: Conflict validation protocol

4.3.5 Negotiation Protocol: Alternating Offers Protocol

This protocol is represented in figure 4.5 using the FIPA notation. Upon deciding to start the negotiation protocol, the agent ag_i checks the message inbox for a negotiation request from the opposing agent, ag_j, which would mean that ag_j initiated the negotiation first. If no request is found, ag_i sends a request to ag_j, and waits for a response. If ag_j accepts to take part in the negotiation, ag_i then calculates a set of prices. Next, ag_i sends that proposal to ag_j, who analyzes it, and either: (i) accepts it, if it meets the utility criterion, (ii) rejects it, if one or more defined limits (issue values, deadline) was breached, or (iii) sends a counterproposal, if the utility criterion was not reached but no limits where broken. After (i) and (ii), the protocol will be closed. If ag_j's choice was (iii), ag_i will face the same three choices, i.e. accept, reject or send a counterproposal. The process ends when both agents reach an agreement (when one of them accepts a specific propose) or when one agent decides to break of negotiation.

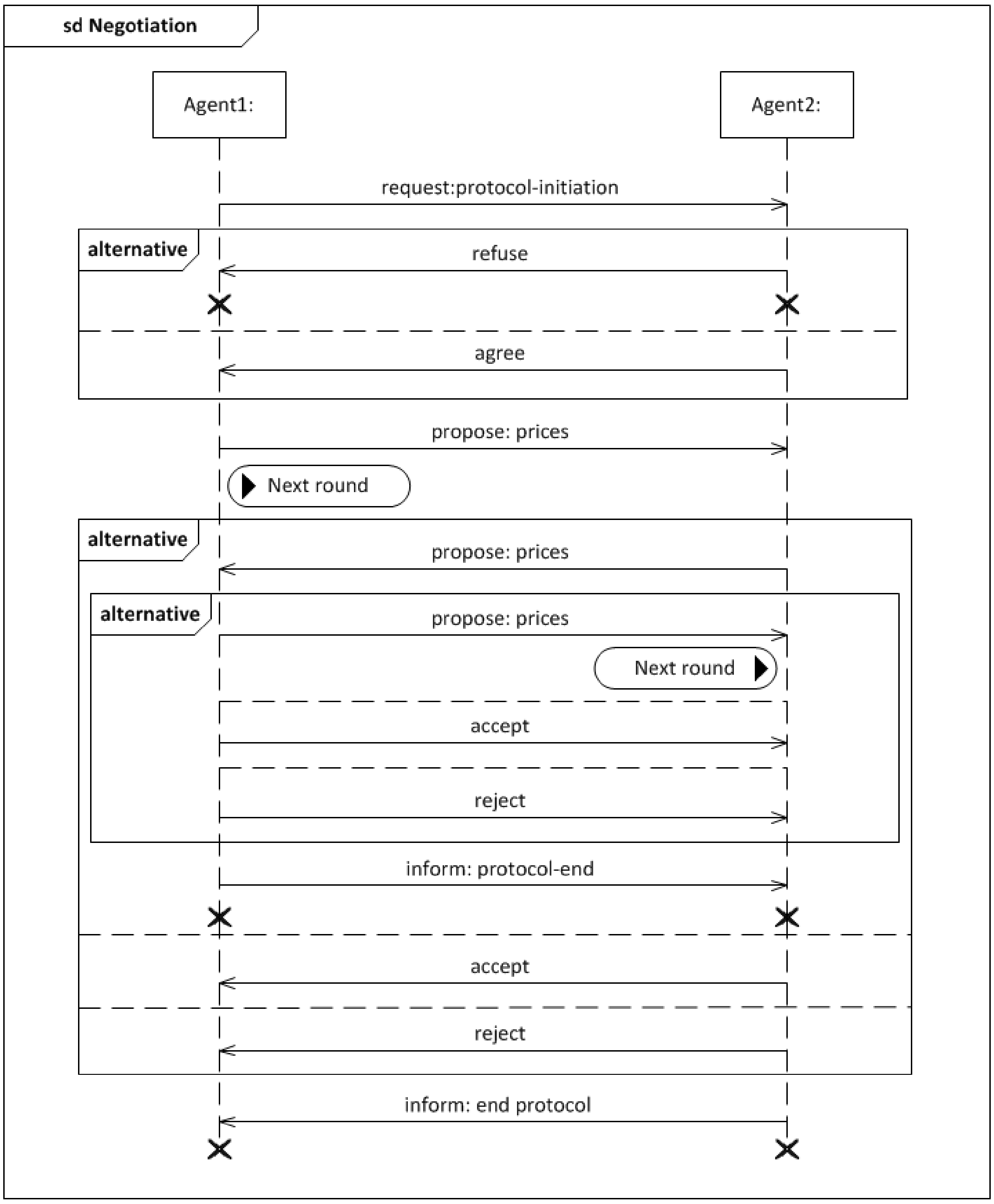


Figure 4.5: Negotiation protocol

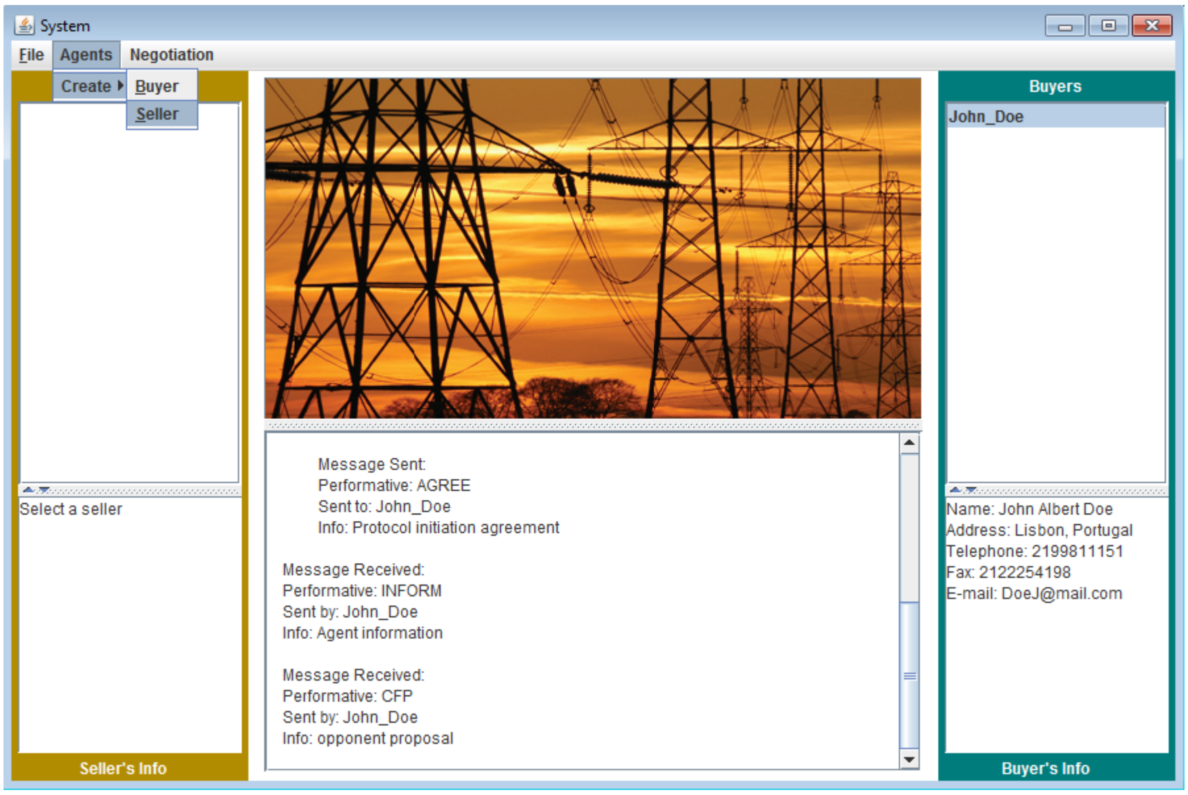
4.4 User-Agent Interface

The simulator NSEM allows users to control several of the agents' actions, as well visualize agents' internal processes and communication history.

4.4.1 Market

The first step is the creation of the market agent. After this, a frame will open on allowing the user to create buyers and sellers (see figure 4.6). To create an agent on the same machine that hosts the market agent, the user simply presses: *agents* $\rightarrow$ *create* $\rightarrow$ *buyer/seller*. A name will be required, and it should correspond to an existing folder containing several required files (plan templates, objectives and beliefs). Following the indication of the agent's name, a few more information is requested for completation of the process.

Alternatively, an agent may be manually created and connected to the market agent's main container (the JADE *environment* where the market agent is located in). When a new agent is connected, the interface will show its name and information on either the left or the right side, according to its type (as shown in figure 4.6). When there is at least one seller and one buyer connected, the user can set them up for negotiation, by pressing: *(while one buyer and one seller are selected)* $\rightarrow$ *negotiation* $\rightarrow$ *set negotiation pair*. After this, both seller and buyer will show their own GUI.



Message Sent:
Performative: AGREE
Sent to: John_Doe
Info: Protocol initiation agreement

Message Received:
Performative: INFORM
Sent by: John_Doe
Info: Agent information

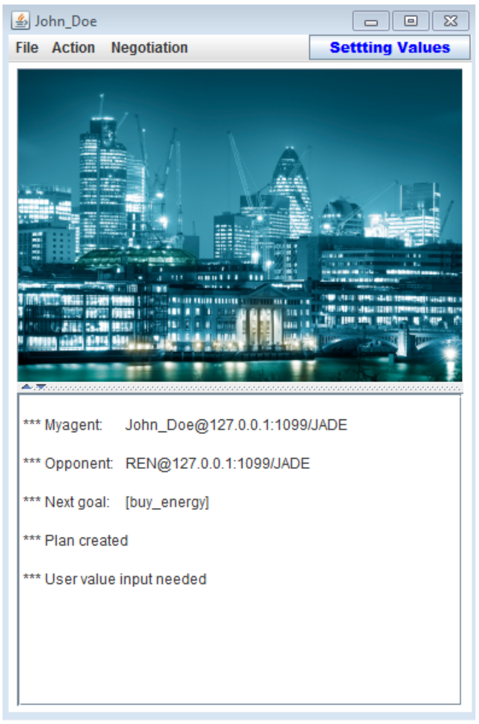
Message Received:
Performative: CFP
Sent by: John_Doe
Info: opponent proposal

Name: John Albert Doe
Address: Lisbon, Portugal
Telephone: 2199811151
Fax: 2122254198
E-mail: DoeJ@mail.com

Figure 4.6: Creating an agent in NSEM

4.4.2 Buyer

The buyer's interface includes a *context* button, i.e. a button appearing on the top right corner in the main window, when it is waiting for a specific action (as seen on figure 4.7). As stated, this agent creates and analyzes a negotiation plan. During this process, it checks whether the user needs to enter a specific value for any variable (by inserting *"ask"* instead of a value in the stored plan template). If that is the case, the user defines one or more values by pressing either: *action* $\rightarrow$ *planning* $\rightarrow$ *setting values*, or the context button. Next, the agent checks for the existence of a conflict by pressing: *action* $\rightarrow$ *conflict* $\rightarrow$ *detect potential conflict*, or again, the context button.

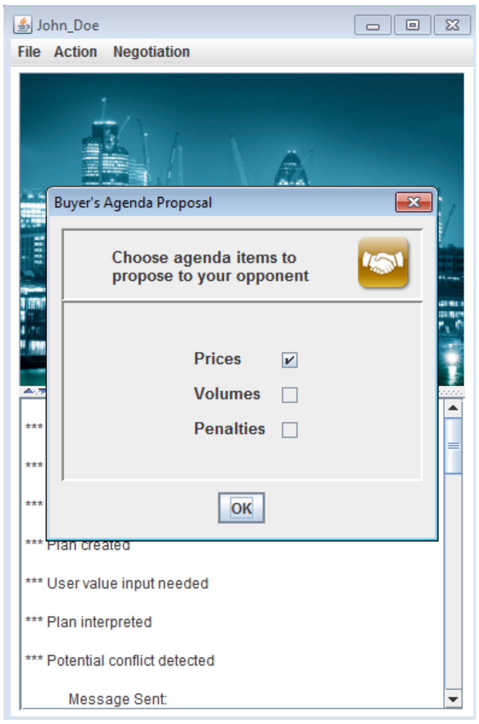


```
*** Myagent:    John_Doe@127.0.0.1:1099/JADE
*** Opponent:   REN@127.0.0.1:1099/JADE
*** Next goal:  [buy_energy]
*** Plan created
*** User value input needed
```

Figure 4.7: The context button in NSEM

At this point, the user can start the process to define the issues at stake, i.e. the agenda, by pressing the context button or choosing: *Negotiation* $\rightarrow$ *Prenegotiation* $\rightarrow$ *Agenda*.

The simulator then checks the primitive plans and gathers all the issues the agent wishes to discuss. It is important to note that the issues not currently supported by the available strategies/protocol are blocked (as shown in figure 4.8). In other words, the current version of NSEM does not support negotiation involving volumes nor penalties.

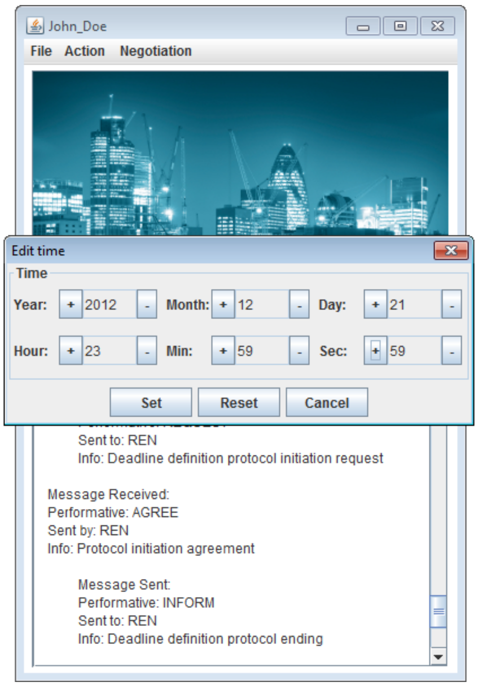


Prices [✓]
Volumes [ ]
Penalties [ ]

```
***
***
***
*** Plan created
*** User value input needed
*** Plan interpreted
*** Potential conflict detected
        Message Sent:
```

Figure 4.8: Agenda definition in NSEM

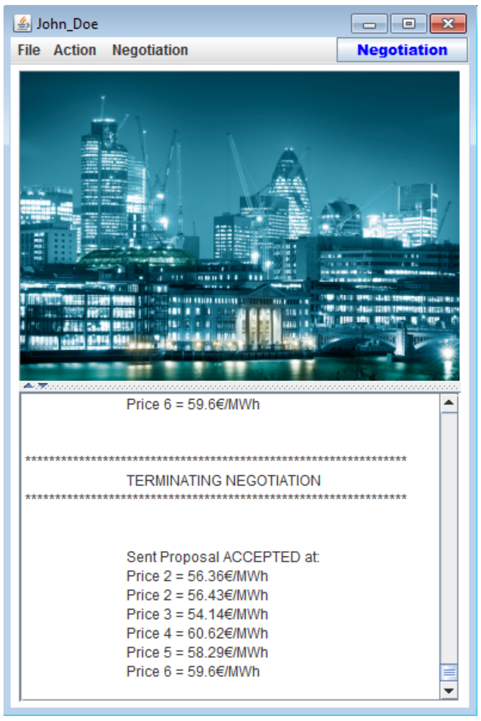
The next step requires the user to perform the remaining prenegotiation tasks by considering the submenu: *Negotiation* → *Prenegotiation* → *Define Targets*, *Define Limits*, *Define Preferences and Strategy*, and finally *Define Deadline* (see figure 4.9). After this, the context button will be visible again, showing the option *Negotiation*. Thus, to start the negotiation process (presented in figure 4.10), the user should either press the context button or choose *Negotiation* → *Negotiation (Process)*.



Year: 2012 Month: 12 Day: 21
Hour: 23 Min: 59 Sec: 59

```
        Sent to: REN
        Info: Deadline definition protocol initiation request

Message Received:
Performative: AGREE
Sent by: REN
Info: Protocol initiation agreement

        Message Sent:
        Performative: INFORM
        Sent to: REN
        Info: Deadline definition protocol ending
```

Figure 4.9: Defining a deadline in NSEM

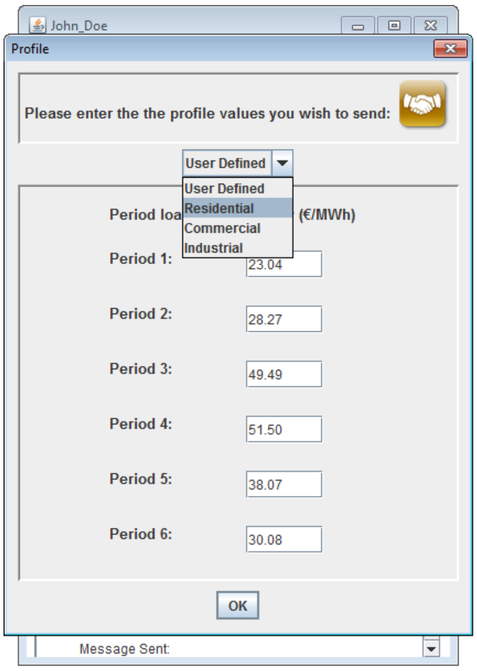


```
                Price 6 = 59.6€/MWh


****************************************************************
                TERMINATING NEGOTIATION
****************************************************************



                Sent Proposal ACCEPTED at:
                Price 2 = 56.36€/MWh
                Price 2 = 56.43€/MWh
                Price 3 = 54.14€/MWh
                Price 4 = 60.62€/MWh
                Price 5 = 58.29€/MWh
                Price 6 = 59.6€/MWh
```

Figure 4.10: Negotiation result in NSEM

During any of the previous tasks, the agent can send its energy profile, i.e. the energy volumes, to the seller agent by pressing: *Action* → *Send Profile*.

The agent's set of intended volumes of electricity are displayed by default. However, there are several profiles predefined, and the user can select any of them, thus updating the volumes displayed (see figure 4.11).



Please enter the the profile values you wish to send:

Period loa (€/MWh)

Period 1:	23.04
Period 2:	28.27
Period 3:	49.49
Period 4:	51.50
Period 5:	38.07
Period 6:	30.08

Message Sent:

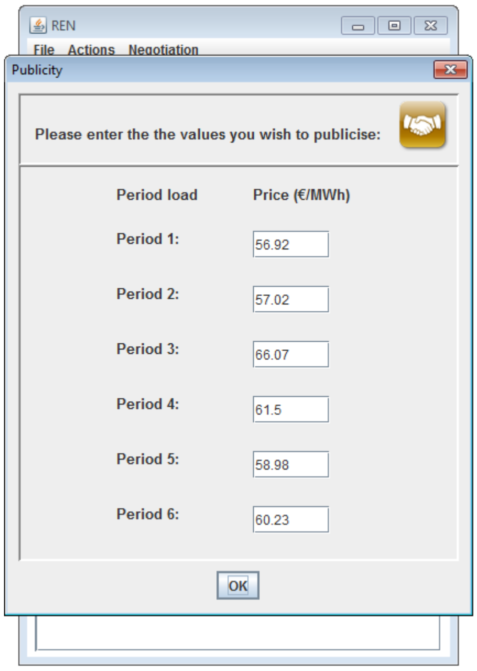
Figure 4.11: Buyer sending a profile in NSEM

Also, it is possible to save the current set of beliefs of the agent to a data file. This action will affect future system sessions with that agent, by changing the agent's initial beliefs. For this feature the user should choose: *File* → *Save Belief Files*.

4.4.3 Seller

After a seller agent shows its interface to the user, it will create the negotiation plan. Similarly to the buyer, if one of the primitive plans features "ask", i.e. indicating that a variable should be defined by the user, the interface will request it. The seller was considered a more passive party, with its prices predefined to publicize them to many potential buyers, before any negotiation begins.

At this point, the user can define the issues in the submenu *Negotiation* → *Prenegotiation*: *Define Targets*, *Define Limits*, *Define Preferences and Strategy* and *Define Deadline* (the user may also choose to let the buyer initiate this protocol). After all these issues have been defined, and the agenda established by a conversation with the buyer, the seller agent can request the start of the negotiation by choosing: *Negotiation* → *Negotiation (Process)*. During any of these tasks, the user can communicate its prices to the buyer, by pressing: *Action* → *Publicize* (as shown in figure 4.12).



Please enter the the values you wish to publicise:

Period load	Price (€/MWh)
Period 1:	56.92
Period 2:	57.02
Period 3:	66.07
Period 4:	61.5
Period 5:	58.98
Period 6:	60.23

Figure 4.12: Seller publicizing a set of prices in NSEM

As stated for the buyer, it is also possible to save the current set of beliefs possessed by the seller to a data file, which will affect future system sessions with that agent. To this end, the user can choose: *File* → *Save Belief Files*.

5

Case Study

Contents

5.1 Introduction

A political agreement between the Portuguese and Spanish governments, signed on 20th January 2004, established the rules for the creation of the Iberian Electricity Market (MIBEL), an energy derivatives market. This agreement determined that energy trading on the Iberian market can be done via the following organized markets:

- day-ahead market (spot) — blocks of energy with physical delivery on the day following trading and mandatory settlement by physical delivery;
- intraday market (spot) — settlement transaction with mandatory physical delivery;
- derivatives market — blocks of energy with physical delivery on the day following trading and with either a physical of financial settlement.

The creation of an Iberian Electricity Market means recognizing a single electricity market between Portugal and Spain, where all participants have indiscriminate access and equal rights and obligations. The market started operating in Portugal on 3rd July 2006 [42].

To test NSEM, this chapter describes a case study including a bilateral negotiation in MIBEL. Two fictitious participants are portrayed, one electricity retailer (seller) and one SME (buyer) that provides legal services. The case study's environment is the electricity retail market, where the seller wants to sell electricity in medium voltage to the buyer. Both agents negotiate the prices of six periods, each referring to a four-hour block on a week-day. The test is not associated with any real life counterpart, but the parameters were chosen by looking up real trading prices associated with a pool market in an attempt to approximate the case study to the real world. Some parameters are not referenced in the chapter, such as the electricity volume limits, because they were not considered in the strategies used by agents.

The Operador del Mercado Ibérico de Energía (OMEL) provides the market results for MIBEL in its web page [43]. The negotiated hourly prices for the day 20th June 2012 were chosen and an average for each four-hour block was calculated (see table 5.1). These prices were used to calculate the target and limit prices for both buyer and seller agents.

Table 5.1: Electricity price set in a pool market divided into a six-period day

Period	Time	Price [EUR/MWh]
1	0:00 - 4:00 h	54.21
2	4:00 - 8:00 h	54.30
3	8:00 - 12:00 h	62.92
4	12:00 - 16:00 h	58.57
5	16:00 - 20:00 h	56.17
6	20:00 - 24:00 h	57.36

5.2 Buyer

The buyer agent creates a negotiation plan by accessing the plan template library and importing the correct plan templates, resulting in a plan with six prices and six volumes of electricity (as explained on section 3.2 and subsection 3.5.1). These prices will be used for conflict detection, and as target (or initial) prices for the negotiation (as shown in table 5.3).

When the buyer agent is initiated, it loads a belief regarding the seller's selling prices (shown in table 5.2). After checking and declaring potential conflict (by comparing the prices in table 5.3 with the prices on table 5.2), the buyer initiates the *Conflict Validation Protocol*, which leads to the declaration of a true conflict. The agent then sends its profile, i.e the volumes of electricity it intends to buy from the seller.

Table 5.2: Buyer's belief regarding the seller's selling prices

Period	Price [EUR/MWh]
1	60.64
2	55.22
3	63.32
4	59.14
5	59.94
6	58.65

Agenda Definition

The buyer's agenda proposal was comprised of prices, volumes and penalty prices (because a conflict was detected and validated, hence the alternative plans were expanded). However, because there's no strategy available to the buyer allowing for volumes and penalty prices, only prices were sent in the proposal. This proposal was then accepted by the seller.

Target and Limit Prices

The target prices, i.e. the prices the buyer wishes to buy energy at, were calculated as 96% of the market price shown in table 5.1, while the maximum prices the buyer is willing to accept for each period, was computed as 104% of the same base price (but see table 5.3).

Table 5.3: Target and limit prices for the buyer agent

Period	Target Price [EUR/MWh]	Limit Price [EUR/MWh]
1	52.04	56.38
2	52.13	56.47
3	60.4	65.44
4	56.23	60.91
5	53.92	58.42
6	55.07	59.65

Target Electricity Volumes

The New York State Electric & Gas (NYSEG) provides on its web page load profiles for weekdays, Saturdays and Sundays. The chosen profiles shared the same time-period with the retrieved prices and its electricity consumption values represent a commercial zone. The volumes were also grouped into four-hour blocks, resulting in six volumes representing the average values for each time block. Table 5.4 shows the target volumes of electricity.

Table 5.4: Target electricity used by the buyer and seller

Period	Target Electricity [MWh]
1	23.04
2	28.27
3	49.49
4	51.5
5	38.07
6	30.08

Strategy and Preferences

In NSEM, the weights of the issues are defined by calculating the ratio between the volume intended to be traded in a specific period and the total volume for all periods (shown in table 5.5). The strategy used by the buyer was *Random Absolute Tit-For-Tat* (see subsection 3.4.2) and the preference model adopted was the *Additive function* (see subsection 3.4.2). The weights are used in the *Random Absolute Tit-For-Tat* strategy when creating the *Random* value to be added to each price in the proposal. The bigger the weight, the smaller the random value can potentially be.

Table 5.5: Weights used by the buyer and seller

Period	Weights
1	0.1
2	0.13
3	0.22
4	0.23
5	0.17
6	0.14

Deadline

The deadline agreed with the seller was one week from the start of the negotiation. However, the strategy chosen does not consider the deadline when generating a proposal.

5.3 Seller

The seller creates a negotiation plan and its associated intention structure by accessing the plan template library (as explained for the buyer on sub-section 3.2) and importing the correct plan templates, resulting in a plan including six prices, which will be used as target prices for the negotiation.

Target and Limit Prices

The target prices the seller desires to sell electricity at, were calculated as 105% of the market price, while the minimum prices the seller will agree to, for each period, were 95% of the market price. The seller is seen as a more acquisitive and resourceful party, hence the slight bigger target and limit prices. The table 5.6 shows these prices. Like the buyer agent, at this point the seller can modify its target prices.

Table 5.6: Seller's target and limit prices

Period	Target Price [EUR/MWh]	Limit Price [EUR/MWh]
1	56.92	51.5
2	57.02	51.59
3	66.07	59.77
4	61.5	55.64
5	58.98	53.36
6	60.23	54.49

Target Electricity Volumes

The seller does not possess specific target electricity volumes as part of its selling plan. As stated, it is expected that the buyer send its electricity profile prior to negotiation. Accordingly, the seller agent will adopt these values as its target volumes to trade. Table 5.4 shows the electricity volumes used by the seller.

Strategy and Preferences

The weights of the issues are also defined by calculating the ratio between the volume intended to be traded in a specific period and the volume total for all periods. The strategy used by the seller was *Random Absolute Tit-For-Tat* (see subsection 3.4.2) and the preference model adopted was the *Additive function* (see subsection 3.4.2). The coefficients for the seller agent are equal to the ones used for the buyer agent. The weights are also identical, since the target electricity volumes are the same. The latter are shown in table 5.5.

5.4 Results and Discussion

The seller initiated the negotiation process, thus sending the first offer. Table 5.7 shows the proposals that were exchanged by the agents during negotiation. Specifically, eight proposals were exchanged the seller accepted a proposal, because the latest received proposal from the buyer (t=7) meant a higher utility value than the next computed proposal (which would be sent to the seller on t=8).

Table 5.8 shows side by side the results from the real pool market, on 20th June 2012 and the prices from the agreement reached in the case study. There is a great deal of similarity between both sets of values, indicating that the results obtained by NSEM, when the agents use the *Random Absolute Tit-For-Tat* strategy, are meaningful. In other words, NSEM can support, at least in part, real-life negotiation of bilateral contracts in electricity markets since the prices of the agreement in bilateral negotiation should not differ drastically from the prices obtained in pool markets.

Table 5.8: Market price and case study results

Period	Market Price [EUR/MWh]	Case Study [EUR/MWh]
1	54.21	54.16
2	54.30	54.25
3	62.92	62.86
4	58.57	58.52
5	56.17	56.12
6	57.36	57.31

Table 5.7: Exchange of proposals towards finding an agreement

Step (t)	Seller	Buyer	Notes
0	Price 1 = 56.92 €/MWh Price 2 = 57.02 €/MWh Price 3 = 66.07 €/MWh Price 4 = 61.50 €/MWh Price 5 = 58.98 €/MWh Price 6 = 60.23 €/MWh	↷	Seller sends first proposal
1	↶	Price 1 = 52.04 €/MWh Price 2 = 52.13 €/MWh Price 3 = 60.40 €/MWh Price 4 = 56.23 €/MWh Price 5 = 53.92 €/MWh Price 6 = 55.07 €/MWh	Buyer sends a counterproposal
2	Price 1 = 55.84 €/MWh Price 2 = 55.93 €/MWh Price 3 = 64.81 €/MWh Price 4 = 60.33 €/MWh Price 5 = 57.86 €/MWh Price 6 = 59.08 €/MWh	↷	Seller sends a counterproposal
3	↶	Price 1 = 52.91 €/MWh Price 2 = 53.00 €/MWh Price 3 = 61.41 €/MWh Price 4 = 57.17 €/MWh Price 5 = 54.82 €/MWh Price 6 = 55.99 €/MWh	Buyer sends a counterproposal
4	Price 1 = 54.97 €/MWh Price 2 = 55.06 €/MWh Price 3 = 63.80 €/MWh Price 4 = 59.39 €/MWh Price 5 = 56.96 €/MWh Price 6 = 58.16 €/MWh	↷	Seller sends a counterproposal
5	↶	Price 1 = 53.60 €/MWh Price 2 = 53.69 €/MWh Price 3 = 62.22 €/MWh Price 4 = 57.92 €/MWh Price 5 = 55.54 €/MWh Price 6 = 56.72 €/MWh	Buyer sends a counterproposal
6	Price 1 = 54.28€/MWh Price 2 = 54.37€/MWh Price 3 = 62.99€/MWh Price 4 = 58.64€/MWh Price 5 = 56.24€/MWh Price 6 = 57.43€/MWh	↷	Seller sends a counterproposal
7	↶	Price 1 = 54.16 €/MWh Price 2 = 54.25 €/MWh Price 3 = 62.86 €/MWh Price 4 = 58.52 €/MWh Price 5 = 56.12 €/MWh Price 6 = 57.31 €/MWh	Buyer sends a counterproposal
8	**ACCEPT**	⇨	Seller accepts counterproposal from t=7

6

Conclusions and Future Work

Contents

6.1 Discussion

Multi-agent systems allow for the simulation and analysis of real-life complex domains. Accordingly, the work in this dissertation used agent technology to develop the simulator NSEM (Negotiation Simulator for the Electricity Market). At its heart, NSEM is a MAS allowing to create autonomous agents capable of generating plans, detecting and validating social conflicts, and negotiate mutually beneficial agreements.

For the purposes of this dissertation, the simulator was adapted to represent the negotiation of bilateral contracts in the Electricity Market (EM), which was set as the first objective of the work. NSEM's use of BDI agents to plan actions, detect conflicts, complete several prenegotiation tasks, and iteratively trade proposals to reach agreements, fulfill the second and third objectives of this dissertation. The last objective was met with the development of a case study, showing that NSEM can help negotiating bilateral contracts in energy markets.

Let's now look back at the research questions presented in section 1.2.

1. *How to integrate an autonomous agent's individual behavior (in particular, the capability to plan actions) with its social behavior (the capability to negotiate contracts) ?*

The key concept for this integration is social conflict: conflict is the focal point of interaction, and the driving force of negotiation. The implementation of conflict detection and validation procedures allows NSEM's agents to plan and execute actions towards achieving their design objectives.

2. *Successful human negotiators often consider prenegotiation, i.e. planning and preparation for negotiation, a crucial factor for a successful negotiation outcome. How to develop software agents able to effectively plan and prepare for negotiation?*

The agents in NSEM execute several prenegotiation tasks, notably: definition of the issues in the agenda, definition of target and limit values for each issue, definition of the preferences, and selection of a strategy. All these tasks allow agents to be better prepared for the actual negotiation phase, granting an awareness of the negotiation status quo — how close the agent is to an agreement; is a received proposal more beneficial than a proposal the agent is about to send; is the agent considering which items are more important; should the agent change its strategy; how long to reach the deadline; is the agent at a point where the issues' values are no longer minimally acceptable.

3. *How to develop software agents capable of negotiating bilateral contracts in the EM, with prenegotiation on its foundation?*

The implemented procedures to manage the actual negotiation tasks, i.e. proposal creation, proposal analysis, proposal submission, strategy implementation, and general negotiation decision-making methods, consider directly the result of the aforementioned prenegotiation tasks, making them a true basis for actual negotiation.

As stated previously, existing EM simulators place emphasis on the negotiation process, giving little attention to prenegotiation tasks. They may consider some isolated prenegotiation tasks, such as the definition of limit and target values, but no attempt is normally made to develop prenegotiation plans. Furthermore, the implementation of an individual BDI structure for the agents is often disregarded, and conflict or any other possible driving force for negotiation is ignored.

Over the last few years, the focus has been on actual negotiation, notably trading protocols and negotiation strategies. NSEM focuses on prenegotiation, addressing a field that has been neglected. Furthermore, NSEM's modular structure allows for future extensions related to the three main negotiation phases.

6.2 Future Work

For future work, we suggest the following additions and improvements to NSEM:

1. **Homogenous Agents –** A deeper analysis of the ultimate goals of NSEM should be performed. Accordingly, it may prove useful to make both the buyer and the seller agents similar in terms of capabilities (e.g., conflict detection and validation).

2. **Agenda –** During prenegotiation, a more complex agenda definition protocol should be developed, for instance, similar to the negotiation protocol (alternating offers protocol).

3. **Strategies –** The actual negotiation phase should be expanded to include more negotiation strategies, specifically strategies involving electricity volumes, and issues such as price penalties and a deadline. More models for proposal evaluation should be adopted and implemented (currently, NSEM considers one model).

4. **User-Interface –** The user-machine interface is one of the most important components of NSEM, and its quality should be assured. Thus, exception/error handling should be considered and the GUI should be tested for usability, granting a better and easier experience to the user. Also, NSEM would benefit from the inclusion of different tariffs, including an hourly rate.

5. Networking – While the application allows for network connections, i.e. different agents can run in different computers, whilst connecting to one computer running the market agent, the environment specifications (such as network environments and firewall settings) have not been sufficiently tested to establish this feature.

NSEM should be adapted and tested in different domains and business environments. Noticeably, NSEM is currently being adapted to a supply chain negotiation environment, and tested by considering a published case study [44]. A supply chain is a network of facilities that performs the functions of acquisition of raw materials from suppliers, transformation of these materials into final products, and the delivery of these products to customers. In future software iterations, it may prove positive to make both buyer and seller equal in terms of features (such as conflict detection), but a deeper analysis of ultimate goals for the simulator should be performed.

Bibliography

1. **Conte, R. and Castelfranchi, C.** *Cognitive and Social Action.* New York : Garland Science, 1995. 978-1857281866.
2. *Multiagent Systems.* **Sycara, K.** 2, Menlo Park : AAAI Press, 1998, AI magazine, Vol. 19, pp. 79-92. 0738-4602.
3. *A Roadmap of Agent Research and development.* **Jennings, N., Sycara, K. and Wooldridge, M.** [ed.] Jeffrey S. Rosenschein. Heidelberg : Springer Science+Business Media, 1998, Autonomous Agent and Multi-agent Systems, Vol. 1, pp. 7-38. 1387-2532.
4. **Bond, A. and Gasser, L.** *Readings in Distributed Artificial Intelligence.* [ed.] L. Gasser. Burlington : Morgan Kaufmann Publishers, 1988. 978-0934613637.
5. **Lopes, F.** *Negociação entre Agentes Computacionais Autónomos.* [thesis] Lisbon : Instituto Superior Técnico, 2004.
6. *Negotiation among autonomous computational agents: principles, analysis and challenges.* **Lopes, F., Wooldridge, M. and Novais, A.** 2, Norwell : Kluwer Academic Publishers, 2009, Artificial Intelligence Review, Vol. 29, pp. 1-44. 0269-2821.
7. Intelligent Agents. [ed.] G Weiss. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence.* Cambridge : The MIT Press, 1999, pp. 27-72.
8. **Russell, S. and Norving, P.** *Artificial Intelligence: A Modern Approach.* Borough : Prentice Hall, 2009. 978-0136042594.
9. *Intelligent Agents: Theory and Practice.* **Wooldridge, M. and Jennings, N.** [ed.] Parsons S. McBurney,. 2, Cambridge : Cambridge University Press, 1995, Knowledge Engineering Review, Vol. 10, pp. 115-152.
10. **Bratman, M.** *Intention, Plans, and Practical Reason.* Cambridge : Cambridge University Press, 1999. 978-1575861920.
11. *Industrial deployment of multi-agent technologies: review and selected case studies.* **Pechoucek, M. and Marik, V.** 3, New York : Springer, 2008, Autonomous Agents and Multi-Agent Systems, Vol. 17, pp. 397-431.
12. **Galliers, J.** "The positive role of conflict in cooperative multi-agent systems", Decentralized AI. [book auth.] Y Demazeau and J. Muller. *Decentralized A.I.: proceedings of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World.* Amsterdam : Elsevier Science Ltd, 1990, pp. 33-46.
13. *Agents that reason and negotiate by arguing.* **Parsons, S., Sierra, C. and Jennings, N.** 3, Oxford : Oxford University Press, 1998, Journal of Logic and Computation, Vol. 8, pp. 261-292.
14. *A negotiation model for autonomous computational agents: Formal description and empirical evaluation.* **Lopes, F., et al.** 3-4, Lansdale : IOS Press, 2002, Journal of Intelligent and Fuzzy Systems, Vol. 12, pp. 195-212. 1064-1246.
15. **Lopes, F., Sousa, J. and Coelho, H.** *Negotiation and Risk Management in Multi-Agent Energy Markets.* [Document] Lisbon : s.n., 2010.
16. **Pruitt, D., Kim, K. and Rubin, J.** *Social conflict: escalation, stalemate and settlement.* New York : McGraw-Hill, 2003. 978-0072855357.

17. **Pruitt, D.** *Negotiation behavior.* New York : Academic Press, 1981. 978-0125662505.

18. *Negotiation and mediation. In: Rosenzweig M, Porter L (eds.).* **Carnevale, P. and Pruitt, D.** 1992, Annual review of psychology, Vol. 43, pp. 531-582.

19. **Lewicki, R., Barry, B. and Saunders, D.** *Negotiation.* 6. New York : McGraw Hill, 2009. 978-0071263641.

20. **Raiffa, H.** *The art and science of negotiation.* Cambridge : Harvard University Press, 1985. 978-0674048133.

21. **Kelley, H.** A classroom study of dilemmas in interpersonal negotiations. [ed.] K. Archibald. *Strategic intervention and conflict.* Berkeley : University of California Press, 1966, pp. pp 49-73.

22. **Michaels, R.** *Vertical Integration and the Restructuring of the U.S. Electricity Industry.* [Document] Fullerton : s.n., 2004. Vertical Integration and the Restructuring of the U.S. Electricity Industry.

23. **Kirschen, D. and Strbac, G.** *Fundamentals of Power System Economics.* Hoboken : John Wiley & Sons Ltd, 2004. 978-0470845721.

24. **Pereira, I.** *Sistema Multi-Agente para Apoio à Negociação em Mercados de Electricidade.* [Dissertation] Vila Real : Universidade de Trás-os-Montes, 2004.

25. *The Open Agent Architecture: A Framework for Building Distributed Software Systems.* **Martin, D., Cheyer, A. and Moran, D.** 1-2, Menlo Park : Taylor & Francis, 1999, Applied Artificial Intelligence: An International Journal, Vol. 13, pp. 91-128. 0883-9514.

26. *Experiences creating three implementations of the repast agent modeling toolkit.* **North, M., Collier, N. and Vos, J.** 1, New York : ACM, January 2006, ACM Transactions on Modeling and Computer Simulation (TOMACS), Vol. 16, pp. 1-25. 1049-3301.

27. **Bellifemine, F., et al.** *JADE: A White Paper.* [Document] September 2003.

28. **Bellifemine, F., Claire, G. and Greenwood, D.** *Developing Multi-Agent Systems with JADE.* Hoboken : John Wiley & Sons, 2007. 978-0-470-05747-6.

29. *Agent-Based Simulation of Electricity Markets: A Survey of Tools.* **Zhou, Z., Chan, W. and Chow, J.** 4, Heidelberg : Kluwer Academic Publishers, 2007, Artificial Intelligence Review, Vol. 28, pp. 305-342. 0269-2821.

30. *Modeling hydro power plants in deregulated electricity markets: Integration and application of EMCAS and VALORAGUA.* **Thimmapuram, P., et al.** Lisbon : s.n., 2008. pp. 1-6. 978-1-4244-1743-8.

31. **Faruqui, A. and Eakin, K.** *Electricity Pricing in Transition.* New York : Springer, 2002. 978-0792376002.

32. *A Web-Based Platform for Experimental Investigation of Electric Power Auctions.* **Zimmerman, R., et al.** 3-4, Amsterdam : Elsevier Science Publishers B. V., January 1999, Decision Support Systems - Special issue on restructuring the electric power businessa new paradigm for reducing regulation, Vol. 24. 0167-9236.

33. *The AMES Wholesale Power Market Test Bed.* Iowa State University. [Online] Iowa State University. [Cited: July 09, 2012.] http://www1.umn.edu/. http://www2.econ.iastate.edu/tesfatsi/AMESMarketHome.htm#FERC2003.

34. *MASCEM: Electricity Markets Simulation with Strategic Agents.* **Vale, Z., et al.** 2, s.l. : IEEE Computer Society, 2011, Intelligent Systems, IEEE, Vol. 26, pp. 9-17. 1541-1672.

35. **Praça, I., et al.** Virtual Power Producers Integration Into MASCEM. [book auth.] L. Matos, et al. *Establishing the Foundation of Collaborative Networks.* New York : Springer, 2007, Vol. 243, pp. 291-298.

36. Cambridge Advanced Learners Dictionary. *Cambridge Dictionaries Online.* [Online] Cambridge University Press. [Cited: August 3, 2012.] http://dictionary.cambridge.org/dictionary/british/.

37. **Martin, M. and Rubinstein, A.** *Bargaining and Markets.* Bradford : Emerald Group Publishing Limited, 1990. 978-0125286329.

38. **Goodwin, P. and Wright, G.** *Decision Analysis for Management Judgment.* Hoboken : John Wiley & Sons, 2004. 978-0470861080.

39. *Negotiation Decision Functions for Autonomous Agents.* **Faratin, P., Sierra, C. and Jennings, N.** 3 - 4, Amsterdam : Elsevier Science B.V., 1998, Robotics and Autonomous Systems, Vol. 24. 0921-8890.

40. *FIPA ACL Message Structure Specification.* **FIPA TC Communication.** Geneva : s.n., 2002. FIPA TC Communication. SC00061G.

41. **Foundation for Intelligent Physical Agents.** *FIPA Modeling: Interaction Diagrams.* [Document] Geneva : FIPA TC Modeling, 2003.

42. **MIBEL Regulatory Council.** *Description of the operation of MIBEL.* [Document] November 2009.

43. Omel Mercados. *Omie.* [Online] OMEL AGENCY SECURITIES MARKETS, SAU. [Cited: September 2, 2012.] http://www.omelmercados.es/omel-mercados/.

44. **Lopes, F. and Coelho, H.** *Strategic and tactical behaviour in automated negotiation.* New Orleans : Spring, 2010. pp. 35-63. Vol. 4. 0974-0635.

A

Java Code & XML: The Buyer

This appendix presents part of the source code of the buyer agent, namely various methods implementing its life-cycle and interaction behavior. Also, this appendix presents part of the Plan Template Library of the buyer agent, namely partial composite and primitive plan templates.

A.1 Buyer's State Management Method

The following method, from *buyer.java*, shows the buyer and its different internal phases:

```
public void executePhase(int phase) {

    this.phase = phase;
    switch (phase) {

        case 0:
            //Ask personal info to send to market agent;
            input_gui.askPersonalInfo();
            //read belifs about itself in file:
            //Agent Data\%agent-name%\beliefs_%agent-name%.xml;
            readBeliefs(getLocalName());
            //add behaviour to contact market agent and await
            //an opponent attribution;
            addBehaviour(new AgentPairingProtocol());
            break;

        case 1:
            //read belifs about opponent in file:
            //Agent Data\%opponent-name%\beliefs_%opponent-name%.xml;
            readBeliefs(getOpponent().getLocalName());
            setup_negotiation_strategies();
            //initiate GUI;
            gui = new BuyerGui(this);
            //Enable necessary buttons;
            gui.guiEnableButtons(1);
            //Read file Agent Data\%agent-name%\objectives.xml get
            //objectives list;
            //Read file Agent Data\%agent-name%\plan_templates.xml
            //create plan with
            //  a plan template identical to the first item in the
```

```
                    //objective list;
                    //Plan interpretation;
                    PlanCreationAndInterpretation();
gui.updateLog2("*** Plan interpreted");
                    //Adds two belifs to the agent’s belief system, one for the
                    //prices inteded to buy at, and one for the intended volumes;
                    createPriceVolumesBeliefString();
                    break;

               case 2:
                    //If during the intrepretation of the plan in phase 1, the
                    //simulator sees that arguments are requires to procced
                    //for one of the primitive plans (by using a ’ask’ instead
                    //of the value in the plan templates) the simulator will
                    //wait for the user to initiate value setting, after those
                    //values are set, it will go to phase 3;
                    gui.guiEnableButtons(2);
                    gui.updateLog2("*** User value input needed");
                    break;

               case 3:
                    //Wait for user to initiate potential conflict detection;
                    gui.guiEnableButtons(3);
                    break;

               case 4:
                    //If the agent has a belief regarding the opponent’s selling
                    //price intention, it will
                    //  go for phase 5, otherwise it will add a behaviour to ask
                    //the opponent for the prices
                    //  and then go to phase 5;
                    gui.guiEnableButtons(4);
                    if (!opponentPricesVolumesExist()) {
                         addBehaviour(new beliefRequestProtocol(getOpponent().
                                   getLocalName(), "prices"));
                    } else {
                         executePhase(5);
                    }
                    break;
```

```
case 5:
    //The agent will look for potential conflict between his
    //intended buying prices
    //  and his belief about the seller's intended selling prices;
    //If potential conflict is found, and prices_received_recently
    //is true, it will
    //  go to phase 7, if prices_received_recently is
    //false, phase 6;
    detectConflict();
    break;

case 6:
    //The agent adds a behavior to validate the conflict
    //with the source,
    //  the opponent;
    addBehaviour(new conflictValidationProtocol());
    break;

case 7:
    //If the conflict was detected, the alternate plans are
    //expanded here;
    planner.createPlanAlternative(0);
    gui.updateLog2("*** Alternate plans expanded");
    executePhase(8);
    break;

case 8:
    //Wait for user to initiate agenda definition;
    gui.guiEnableButtons(8);
    break;

case 9:
    //Initiates the agenda definition protocol
    gui.guiEnableButtons(9);
    addBehaviour(new agendaDefinitionProtocol());
    break;

case 10:
```

```
                //Wait for the user to define all the fields in the
                //prenegotiation tab
                gui.guiEnableButtons(10);
                break;

            case 11:
                //wait for the user to initiate the negotiation
                gui.guiEnableButtons(11);
                break;



            case 12:
                //Initiates negotiation
                BuyerNegotiation market_buyer = new BuyerNegotiation(this);
                market_buyer.purchase("", ArrayListToArray(prices_target),
                        ArrayListToArray(prices_limit),
                        ArrayListToArray(volumes_target),
                        negotiation_strategy, deadline);
                break;
        }
    }
```

A.2 Belief Management methods

The following methods are used to manage the agent's own beliefs and the beliefs about opposing agents

```
...

    private HashMap<String, ArrayList<String>> beliefs_about_others = new HashMap();
    private ArrayList<String> beliefs_about_myagent = new ArrayList<String>();

...

    protected void addBelif(String name, String belief) {
        //while adding a belief, if one already exists with he same
        //description, the new one replaces it;
        if (name.equals("myagent") || name.equals(getLocalName())) {
            if (searchBelief(getLocalName(), belief.split(";")[1]) != null) {
                removePartialBelief(getLocalName(), belief.split(";")[1]);
            }
            getBelifsAboutMyAgent().add(belief);

        } else if (getBelifsAboutOthers().containsKey(name)) {

            if (searchBelief(name, belief.split(";")[1]) != null) {
                removePartialBelief(name, belief.split(";")[1]);
            }

            ArrayList<String> list = getBelifsAboutOthers().get(name);
            list.add(belief);
            getBelifsAboutOthers().put(name, list);

        } else {
            ArrayList<String> list = new ArrayList<String>();
            list.add(belief);
            getBelifsAboutOthers().put(name, list);
        }
    }


    protected boolean beliefExists(String name, String belief) {
        //checks if the belief is owned by the agent (must be completely equal,
        //not just the description);
        if (name.equals("myagent") || name.equals(getLocalName())) {
            for (int i = 0; i < getBelifsAboutMyAgent().size(); i++) {
                if (getBelifsAboutMyAgent().get(i).equals(belief)) {
                    return true;
                }
            }

        } else if (getBelifsAboutOthers().containsKey(name)) {
            for (int i = 0; i < getBelifsAboutOthers().get(name).size(); i++) {
                if (getBelifsAboutOthers().get(name).get(i).equals(belief)) {
                    return true;
                }
            }
        }
        return false;
    }


    protected String searchPartialBelief(String name, String part_belief) {
        //checks if any beliefs currently owned by the agent contains the
        //segment in part_belief, if so returns it;
        if ((name.equals("myagent") || name.equals(getLocalName())) &&
                !getBelifsAboutMyAgent().isEmpty()) {
            for (int i = 0; i < getBelifsAboutMyAgent().size(); i++) {
                if (getBelifsAboutMyAgent().get(i).contains(part_belief)) {
                    return getBelifsAboutMyAgent().get(i);
                }
            }

        } else if (getBelifsAboutOthers().containsKey(name)) {
            for (int i = 0; i < getBelifsAboutOthers().get(name).size(); i++) {
                if (getBelifsAboutOthers().get(name).get(i).
                        contains(part_belief)) {
                    return getBelifsAboutOthers().get(name).get(i);
                }
            }
        }
        return null;
    }


    protected String searchBelief(String name, String belief_header) {
        //checks if the belief is owned by the agent (must be completely equal)
        //and returns it;
        if ((name.equals("myagent") || name.equals(getLocalName())) &&
                !getBelifsAboutMyAgent().isEmpty()) {
            for (int i = 0; i < getBelifsAboutMyAgent().size(); i++) {
                if (getBelifsAboutMyAgent().get(i).split(";")[1].
                        equals(belief_header)) {
                    return getBelifsAboutMyAgent().get(i);
                }
            }

        } else if (getBelifsAboutOthers().containsKey(name)) {
            for (int i = 0; i < getBelifsAboutOthers().get(name).size(); i++) {
                if (getBelifsAboutOthers().get(name).get(i).split(";")[1].
                        equals(belief_header)) {
                    return getBelifsAboutOthers().get(name).get(i);
                }
            }
        }
        return null;
    }



    protected void removeBelief(String name, String belief) {
        //removes the exact belief received in the string;
        if (name.equals("myagent") || name.equals(getLocalName())) {
            for (int i = 0; i < getBelifsAboutMyAgent().size(); i++) {
                if (getBelifsAboutMyAgent().get(i).equals(belief)) {
                    getBelifsAboutMyAgent().remove(i);
                }
            }

        } else if (getBelifsAboutOthers().containsKey(name)) {
            for (int i = 0; i < getBelifsAboutOthers().get(name).size(); i++) {
                if (getBelifsAboutOthers().get(name).get(i).equals(belief)) {
                    getBelifsAboutOthers().get(name).remove(i);
                    if (getBelifsAboutOthers().get(name).isEmpty()) {
                        getBelifsAboutOthers().remove(name);
                        return;
                    }
                }
            }
        }
    }



    protected void removePartialBelief(String name, String part_blief) {
        //removes a belief that partially contain the received belief
        if (name.equals("myagent") || name.equals(getLocalName())) {
            for (int i = 0; i < getBelifsAboutMyAgent().size(); i++) {
                if (getBelifsAboutMyAgent().get(i).contains(part_blief)) {
                    getBelifsAboutMyAgent().remove(i);
                }
            }
        } else if (getBelifsAboutOthers().containsKey(name)) {
            for (int i = 0; i < getBelifsAboutOthers().get(name).size(); i++) {
                if (getBelifsAboutOthers().get(name).get(i).
                        contains(part_blief)) {
                    getBelifsAboutOthers().get(name).remove(i);
                    if (getBelifsAboutOthers().get(name).isEmpty()) {
                        getBelifsAboutOthers().remove(name);
                        return;
                    }
                }
            }
        }
    }
```

A.3 Conflict Validation Protocol

The following JADE behavior is the Conflict Validation Protocol, from the buyer's point of view:

```
    private class conflictValidationProtocol extends Behaviour {

        ACLMessage msg_rcv;
        ACLMessage reply;
        MessageTemplate mt = MessageTemplate.and(
                MessageTemplate.MatchOntology("market_ontology"),
                MessageTemplate.MatchProtocol("conflict_validation"));
        private int step = 0;
        String opponent_price_belief = "";
        private boolean conflict = true;
```

```
@Override
public void action() {
     switch (step) {

          case 0:
               ACLMessage msg_init = new ACLMessage(ACLMessage.REQUEST);
               msg_init.setContent("init_conflict_validation");
               msg_init.setOntology("market_ontology");
               msg_init.setProtocol("no_protocol");
               msg_init.addReceiver(getOpponent());
               msg_init.setReplyWith(String.valueOf(
                         System.currentTimeMillis()));

               // Sends a request for conflict validation
               send(msg_init);
               printMessage(msg_init, false, "Conflict validation"
                         + " protocol initiation request");

               mt = MessageTemplate.and(MessageTemplate.and(
                         MessageTemplate.MatchOntology("market_ontology"),
                         MessageTemplate.MatchProtocol(
                         "conflict_validation")),
                         MessageTemplate.MatchInReplyTo(
                         msg_init.getReplyWith()));

               step = 1;
               block();
               break;

          case 1:

               //receives an agree for conflict validation
               msg_rcv = myAgent.receive(mt);
               if (msg_rcv != null) {
                    if (msg_rcv.getPerformative() == ACLMessage.AGREE) {
                         printMessage(msg_rcv, true, "Protocol "
                                   + "initiation agreement");
                         opponent_price_belief = searchBelief(getOpponent().
                                   getLocalName(), "prices");
                    reply = msg_rcv.createReply();
                    reply.setPerformative(ACLMessage.QUERY_IF);
                    reply.setContent(opponent_price_belief);

                    //sends a query-if to check if
                    //prices belief is correct
                    send(reply);
                    printMessage(reply, false, "Seller prices for"
                            + " confirmation");

                    mt = MessageTemplate.and(MessageTemplate.and(
                            MessageTemplate.MatchOntology(
                            "market_ontology"),
                            MessageTemplate.MatchProtocol(
                            "conflict_validation")),
                            MessageTemplate.MatchInReplyTo(
                            reply.getReplyWith()));

                    step = 2;

                } else if (msg_rcv.getPerformative()
                        == ACLMessage.REFUSE) {
                    step = 4;
                }
            }
            block();
            break;

        case 2:

            //Receives either a confirm or a disconfirm for the
            //prices belief
            msg_rcv = myAgent.receive(mt);

            if (msg_rcv.getPerformative() == ACLMessage.CONFIRM) {
                printMessage(msg_rcv, true, "Prices confirmed");
                reply = msg_rcv.createReply();
                reply.setPerformative(ACLMessage.INFORM);
                reply.setContent("conflict_exists");
                //sends an inform that the conflict is validated
                send(reply);
                printMessage(reply, false, "There is a conflict");
                removePartialBelief(msg_rcv.getSender().getLocalName(),
                        "in_potential_conflict");
                addBelif(msg_rcv.getSender().getLocalName(),
                        msg_rcv.getSender().getLocalName()
                        + ";in_conflict");
                step = 4;
                break;
            } else if (msg_rcv.getPerformative()
                    == ACLMessage.DISCONFIRM) {
                printMessage(msg_rcv, true, "Prices disconfirmed");
                reply = msg_rcv.createReply();
                reply.setPerformative(ACLMessage.REQUEST);
                reply.setContent("prices");
                //sends a request for prices
                send(reply);
                printMessage(reply, false, "Opponent prices");
                step = 3;
            }

            mt = MessageTemplate.and(MessageTemplate.and(
                    MessageTemplate.MatchOntology("market_ontology"),
                    MessageTemplate.MatchProtocol(
                    "conflict_validation")),
                    MessageTemplate.MatchInReplyTo(
                    reply.getReplyWith()));
            block();
            break;

        case 3:

            //Receives updated prices
            msg_rcv = myAgent.receive(mt);
            if (msg_rcv.getPerformative() == ACLMessage.INFORM) {
                printMessage(msg_rcv, true, "Seller's prices");
                removePartialBelief(msg_rcv.getSender().getLocalName(),
                        "prices");
                    addBelif(msg_rcv.getSender().getLocalName(),
                            msg_rcv.getContent());
                    if (reDetectConflict()) {
                        reply = msg_rcv.createReply();
                        reply.setPerformative(ACLMessage.INFORM);
                        reply.setContent("conflict_exists");
                        printMessage(reply, false, "Conflict detected");
                    } else {
                        reply = msg_rcv.createReply();
                        reply.setPerformative(ACLMessage.INFORM);
                        reply.setContent("no_conflict_exists");
                        conflict = false;
                        printMessage(reply, false, "No conflict detected");

                    }
                }
                send(reply);
                step = 4;
                break;

            case 4:

                ACLMessage msg_end = new ACLMessage(ACLMessage.INFORM);
                msg_end.setContent("end_conflict_validation");
                msg_end.setOntology("market_ontology");
                msg_end.setProtocol("conflict_validation");
                msg_end.addReceiver(getOpponent());
                send(msg_end);
                printMessage(msg_end, false, "Conflict validation"
                        + " protocol ending");
                step = 5;
                break;
        }
    }

    @Override
    public boolean done() {
        if (step == 5) {
            if (conflict) {
                    executePhase(7);
                } else {
                    executePhase(3);
                }
                return true;
            } else {
                return false;
            }
        }
    }
```

A.4 Stored Plan Templates

The follow excerpt is presented in XML. It represents the negotiation plan template "*buy_energy*″ present in the Plan Template Library (*plan_templates.xml*) from the buyer agent named $John_{D}oe$.

```
<pt>
<header>[buy_energy]</header>
<type>composite</type>
<body>
<body_header>[prices]</body_header>
<body_header>[volumes]</body_header>
<body_header>[extras]</body_header>
</body>
</pt>
```

The follow excerpt is a composite plan template "*prices*″ referenced in the previous plan template $Buy_{E}nergy$:

```
<pt>
<header>[prices]</header>
<type>composite</type>
<body>
<body_header>[price_1,[value,52.04]]</body_header>
<body_header>[price_2,[value,52.13]]</body_header>
<body_header>[price_3,[value,60.40]]</body_header>
<body_header>[price_4,[value,56.23]]</body_header>
<body_header>[price_5,[value,53.92]]</body_header>
```

```
<body_header>[price_6,[value,55.07]]</body_header>
</body>
</pt>
<pt>
```

The follow excerpt is a primitive plan template “$Price''_1$ referenced in the previous plan template *Prices*:

```
<pt>
<header>[price_1,[value,arg1]]</header>
<type>primitive</type>
<body>
<body_variable>[set_price_1,[arg1]]</body_variable>
</body>
</pt>
```

Finally, we have two plan templates with the same header – “extras”. This means only one of these plan templates will be added to the plan during plan selection. The other, will be added as an alternative plan (as detailed in subsection 3.5.2):

```
<pt>
<header>[extras]</header>
<type>composite</type>
<body>
<body_header>[no_extras]</body_header>
</body>
</pt>
```

```
<pt>
<header>[extras]</header>
<type>composite</type>
<body>
<body_header>[penalty_prices]</body_header>
</body>
</pt>
```