Designing and Trusting Multi-Agent Systems for B2B Applications

Rafiul Alam

A Thesis

in

The Concordia Institute

for

Information Systems Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Applied Science (Quality Systems Engineering) at
Concordia University
Montreal, Quebec, Canada

September 2008

Your file Votre référence
ISBN: 978-0-494-45510-4
Our file Notre référence
ISBN: 978-0-494-45510-4

ABSTRACT

Designing and Trusting Multi-Agent Systems for B2B Applications

Rafiul Alam

This thesis includes two main contributions. The first one is designing and implementing *B*usiness-to-*B*usiness (*B2B*) applications using multi-agent systems and computational argumentation theory. The second one is trust management in such multi-agent systems using agents' credibility.

Our first contribution presents a framework for modeling and deploying *B2B* applications, with autonomous agents exposing the individual components that implement these applications. This framework consists of three levels identified by strategic, application, and resource, with focus here on the first two levels. The strategic level is about the common vision that independent businesses define as part of their decision of partnership. The application level is about the business processes, which are virtually integrated as result of this common vision. Since conflicts are bound to arise among the independent applications/agents, the framework uses a formal model based upon computational argumentation theory through a persuasion protocol to detect and resolve these conflicts. Termination, soundness, and completeness properties of this protocol are presented. Distributed and centralized coordination strategies are also supported in this framework, which is illustrated with an online purchasing case study followed by its implementation in Jadex, a java-based platform for multi-agent systems.

An important issue in such open multi-agent systems is how much agents trust each other. Considering the size of these systems, agents that are service providers or customers in a *B2B* setting cannot avoid interacting with others that are unknown or partially known regarding to some past experience. Due to the fact that agents are self-interested, they may jeopardize the mutual trust by not performing the actions as they are supposed to. To this end, our second contribution is proposing a trust model allowing agents to evaluate the credibility of other peers in the environment. Our multi-factor model applies a number of measurements in trust evaluation of other party's likely behavior. After a period of time, the actual performance of the testimony agent is compared against the information provided by interfering agents. This comparison process leads to both adjusting the credibility of the contributing agents in trust evaluation and improving the system trust evaluation by minimizing the estimation error.

Keywords: *Multi-agent systems, B2B applications, argumentation theory, agent communication, dialogue games, persuasion, trust.*

ACKNOWLEDGMENTS

First, I would like to convey my gratitude to my supervisor, Dr. Jamal Bentahar, for being an enthusiastic and genius supervisor. He introduced me to the concept of Multi-Agent System (MAS), the basis of this thesis, and provided me a good source of inspiration. He also helped with his useful suggestions on the presentation of the results. This research would have been impossible without his helps and encouragements.

Many thanks to Dr. Ben Hamza, Dr. Amin Hammmad, Dr. Nizar Bouguila for their guidance and support on my course works and projects. Thanks go to the readers of this thesis, for their thoughtful comments. Also, I would like to thank all the faculty and staff at the Concordia Institute for Information Systems Engineering at Concordia University for providing such a nice academic environment.

I would like to thank my friends in course works and projects, my colleagues in the lab, friends and relatives in Montreal for their continuous help and support in work as well as leisure and also Babak and Maziar for collaborating with the project mentioned in Chapter 5.

Last but not least, my dear parents and my family receive my heartfelt gratitude for their sweetest support and never-ending love. I always feel the warmth of their love and affection especially of my wife Mahreen. This thesis is dedicated to all of them.

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

In this chapter, we explain what initiated our interest into the design and implementation of *B2B* applications using argumentative agents and identify some related technologies. In such open multi-agent systems, before interacting with another agent for any scenario, agents representing businesses need to trust each other. This trust management with credibility is another scope of this thesis. We also specify research problems under consideration, describe our contributions, and present the structure of the thesis.

1.1 Context of Research

Performance and competition challenges are nowadays putting businesses under constant pressure to meet changing requirements. This fuels the need for continuous merge and sometimes re-engineering of business processes, resulting in *B*usiness-to-*B*usiness (*B2B*) applications development. Briefly, a *B2B* application is a set of business processes that make disparate autonomous entities (e.g., departments, businesses) collaborate to achieve a common set of goals. Despite the multiple initiatives on *B2B* applications [50, 54, 59, 61], not much exists in terms of modeling and deploying such applications from intelligent and argumentative-agents perspective. By modeling, we mean identifying all the necessary components that connect assets of independent entities engaged in a *B2B* scenario. By deployment, we mean identifying all the necessary technologies that make the connection of these assets happen effectively. Finally, by argumentation we mean making software and autonomous agents comply with a dialectical process to affirm or disavow the conclusions that these agents wish to reciprocally convey. In a *B2B* scenario, argumentation would broadly mean assisting businesses, through representative agents, engage in intense negotiation and persuasion sessions prior to making any joint decisions. The argumentation capability of an agent representing a business can assist this business in negotiating with its peers during a conflict situation and in collaborating with them to achieve agreements about their strategies. Our research addresses the challenge of using argumentation theory for multi-agent systems to develop *B2B* applications, and a case study is used to illustrate the proposed framework followed by its implementation. This framework is an initiative within the emerging field of developing intelligent systems [48, 46]. The technique we are using in this thesis is different from other techniques proposed in this field such as the Lyee methodology [49].

In this context of open multi-agent systems for *B2B* applications, trust plays a fundamental role. Trust models for multi-agent systems represent a set of trust meta-data to define the trust level of the participating agents [21, 40, 41, 66]. In this thesis, our aim is to develop an efficient trust assessment process. To do so, agents mutually interact and rate each other based on the interactions done (either satisfactory or dissatisfactory). The obtained ratings are accumulated to make the trustworthiness of a particular agent. Inter-agent communication is regulated by protocols (shared amongst agents and thus public) and determined by strategies (internal to agents and thus private). Here, agents are capable of evaluating the trust level of the agents which are not known (or not very well known) by consulting other agents who can provide suggestions about the trustworthiness level of other agents. The idea of consulting with others originates from the fact that agents by nature assess diverse trust levels of an agent depending on their different experiments of direct interaction with that specific target agent.

1.2 Motivations

In order to facilitate agile business and to support dynamic partnership formation, information systems are designed to support interoperability. In particular, interoperation between agent systems and electronic business processes is more interesting because of the benefits that can be achieved from both technologies to accomplish complex goals. Our first motivation is to present a framework, which will address different levels and components of e-business applications by intelligent agents that will reason and make decisions. Levels such as *resource*, *application*, and *strategic* in an e-business setting are connected through vertical relations such as *rely-on* and *run-on-top-of* or horizontal relations such as *interconnectivity*, *composition*, and *collaboration*. The agents in a *B2B* application should be equipped with argumentation capabilities to assist a specific component (i) persuade peers of collaborating, (ii) interact with peers during business process implementation, (iii) resolve conflicts that could impede collaboration, and (iv) track conflict resolution.

To be able to interact flexibly in *B2B* dynamic environments, agents need to use advanced communication mechanisms and to achieve trust. Our second motivation is to find a way to help agents reason about their communicative acts, combine them efficiently for complex interactions and achieve the demanded trust. In order to reach that goal, we propose a framework for agent communication based upon logical rules agents can combine to take part in complex interactions such as negotiation. For trust consideration, the proposed model deals with the classification of agents according to their level of truthfulness, which help agents to learn and decide in a dynamic environment where agents may join and leave the system at their own will.

Providing a formal model with termination, soundness, and completeness properties for resolving potential conflicts between businesses in our integrated model for *B2B* applications is our third motivation. Our final motivation is proving the efficiency of the proposed trust model through simulation using Jadex, a programming platform for intelligent agents. By efficiency we mean that the malicious agents in the environment are detected faster than any other model in the literature.

1.3 Research Questions

The overall research questions we are considering in this thesis are the following:

1. How multi-agent systems can be used to design and deploy *B2B* applications?
2. How agents can play different roles in a *B2B* scenario? How should they develop arguments and resolve conflicts?
3. How an agent can trust another agent in a *B2B* dynamic environment?

4. What kind of architecture do we use? Which platform do we select for *B2B* applications?

1.4 Contributions

The thesis contributions are summarized in three points:

1. Introducing a framework for designing *B2B* applications using argumentative agents that combine multi-agent technology and computational argumentation theory.
2. Implementing the proposed framework within a case study about a purchase-order scenario using Web services.
3. Proposing a trust model for *B2B* applications including service providers and customers and proving its efficiency through experimental results.

1.5 Outline

This thesis is divided into 6 chapters and 2 appendices. Chapter 2 and Chapter 3 are about the state of the art. Chapter 2 introduces multi-agent technology, argumentation theory, negotiation and trust in multi-agent systems. Chapter 3 presents multi-agent programming with some methodologies and platforms. Chapters 4 and 5 are about our main contributions. Chapter 4 includes the design and implementation of *B2B* applications using multi-agent systems where agents are argumentative. Chapter 5 presents the trust evaluation model. Finally, Chapter 6 concludes the thesis by summarizing our contributions and identifying directions for future work. Appendix 1 presents the agent definition file used in Jadex implementation. Appendix 2 provides proofs for some propositions and theorems.

Chapter 2. Multi-Agent Systems: an Overview

A multi-agent system (MAS) is one that consists of a number of agents, which interact with one-another. In the most general case, agents will be acting on behalf of users with different goals and motivations. To successfully interact, they will require the ability to cooperate, coordinate, and negotiate with each other [1, 2]. By definition, this system is composed of multiple interacting intelligent agents and is used to solve problems, which are difficult or impossible for an individual agent or monolithic system to solve. Examples of problems, which are appropriate to multi-agent systems research, include online trading [3], disaster response [4], and modeling social structures [5].

After defining an agent in Section 2.1, we devote Section 2.2 to negotiation mechanism in MASs. Section 2.3 introduces argumentation in negotiation. Section 2.4 addresses the importance and evaluation of trust. Characteristics required for an agent to be learning are discussed in Section 2.5. The reader is referred to the references in each section to obtain more knowledge.

2.1 Definition of an Agent

An agent is a computer system that is capable of independent actions in some environment on behalf of its user or owner (figuring out what needs to be done to satisfy design objectives, rather than constantly being told) in order to meet its design objectives. An intelligent agent is a computer system capable of flexible autonomous actions in some environment. We mean by flexible that agent has the following capabilities:

Reactivity: intelligent agents are able to perceive their environment and response in a timely fashion to changes that occur in order to meet their design objectives.

Pro-activeness: intelligent agents can generate and attempt to achieve goals. They are not driven solely by events, but they can take the initiative.

Social ability: intelligent agents are capable of interacting with other agents (possibly humans) via some kind of agent-communication language in order to satisfy their design objectives [1, 2].

Weiss [1] defines an agent as "*a real or virtual entity which is emerged in an environment where it can take some actions, which is able to perceive and represent partially this environment, which is able to communicate with the other agents and which possesses an autonomous behavior that is a consequence of its observations, its knowledge and its interactions with the other agents*". Figure 2.1 represents the characteristics of an agent.

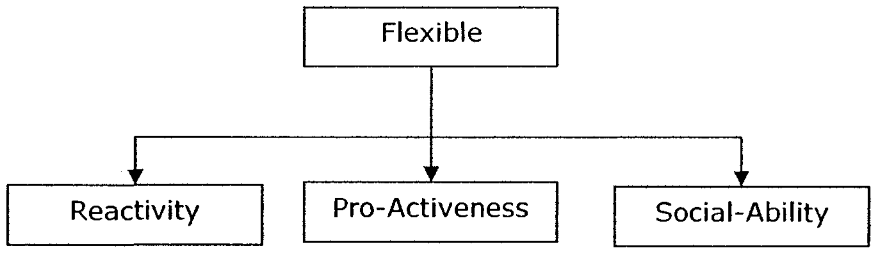


Figure 2.1 Agent characteristics

2.2 Negotiation in Multi-Agent Systems

As a type of interaction, negotiation is gaining increasing prominence in agent computing. By negotiating, agents with conflicting interests, but with a desire to cooperate, try to come to a mutually acceptable agreement on the division of scarce resources [6, 7, 8, 9]. Resources can be money, services, time, commodities etc. Resources are scarce in the sense that competing claims over them cannot be fully satisfied simultaneously. The problem of resource negotiation in a distributed setting is core to a number of applications, particularly the emerging semantic grid computing-based applications such as e-science and e-business. To allow agents to autonomously negotiate with each other, some researchers propose to equip them with argumentation and logical reasoning capabilities [9, 10, 11, 12, 13, 14, 15]. The idea is to use dialogue games as well as the fact that agents should have an argumentative ability to facilitate their communication and negotiation. Dialogue games are rules governing agent interactions by defining pre and post conditions of communicative acts, also called dialogue moves [6, 16, 15, 17].

2.3 Argumentation in Negotiation

There are many ways to classify existing approaches to automated negotiation but we discuss here the three major classes of approaches that suit our purpose in the multi-agent literature.

2.3.1 Game-Theoretic Approaches to Negotiation

Game theory has its roots in the work of Neuman and Morgenstern (1944) [79]. It is a branch of economics (Osborne & Rubinstein, 1994) [80] that studies the strategic interactions between self-interested agents [18]. Game-theory-based negotiation techniques have been widely used in agent systems (Rosenschein & Zlotkin, 1994; Sandholm, 2002b) [81, 82]. The key concepts in this approach to negotiation are:

1. Utility functions;
2. A space of deals;
3. Strategies and negotiation protocols.

The difference between the worth of achieving a goal and the price paid achieving it is defined as utility. Usually, the utilities are given as decision matrices, where an agent looks up a value for a certain action. Using a strategic reasoning, the agent will perform the action with the lowest or highest value. Utility functions represent the prices or costs for activities, in the context of negotiation.

A negotiation protocol defines the rules that govern the negotiation, including the process of termination. Several negotiation protocols can exist in a complex agent-based system. The process of negotiation is described as follows. As an outcome of an interaction for an agent, utility values are built into a payoff matrix, which is shared by both of the parties involved in the negotiation. Each agent chooses a deal, which maximizes its expected utility during offers and counter-offers generated in the process of negotiation. An agent evaluates the other's offer at each step in terms of its own negotiation strategy. The negotiation process might depend on the agent's internal goal of maximizing its utilities, but the decisions are settled on the basis of utility optimization. "*In game-theoretic analysis, researchers usually attempt to determine the optimal strategy by analyzing the interaction as a game between identical participants, and seeking its equilibrium*" [19].

Game-theory based negotiation for multi-agent systems fails to address some crucial issues according to Nwana et al. (1996) [83]:

1. Agents are presumed to be fully rational and acting as utility maximizers using predefined strategies.
2. Each has knowledge of its payoff matrix, and therefore full knowledge of the other agent's preferences. This is certainly unlike the real world where agents only have partial or incomplete knowledge of their own domains. Therefore, this is unrealistic for truly non-benevolent and loosely coupled agents. Further, the payoff matrix can become very large and intractable for a negotiation involving many agents and outcomes.
3. Agents only consider the current state when deciding on their deal; past interactions and future implications are simply ignored.
4. Agents are considered to have identical internal models and capabilities.
5. Much of the work presumes two agents negotiating, though some later work is addressing n-agent negotiation.

2.3.2 Heuristic-based Approaches to Negotiation

A number of heuristic approaches have emerged to address some of the limitations of game-theoretic approaches mentioned above. Heuristics can be seen as rules of thumb that produce good enough (rather than optimal) outcomes and are often produced in contexts with more relaxed assumptions about agents' rationality and resources. Empirical testing and evaluation are required to support particular heuristics (e.g. Faratin, 2000; Kraus, 2001) [84, 85]. Examples of this approach are in [18]. Though heuristic methods can overcome some of the shortcomings of game-theoretic approaches, they also have a number of disadvantages (Jennings et al., 2001) [86]. The models often lead to outcomes that are sub-optimal because they adopt an approximate notion of rationality and because they do not examine the full space of possible outcomes. Furthermore, it is very difficult to predict precisely how the system and the constituent agents will behave. As a result, the models need extensive evaluation through simulations and empirical analysis.

2.3.3 Argumentation-based Approaches to Negotiation

Argumentation-based approaches to negotiation attempt to overcome the above limitations by allowing agents to exchange additional information, or to pursue about their beliefs and other mental attitudes during the negotiation process. In negotiation, an argument is a piece of information that may allow an agent to justify its negotiation stance, or influence another agent's negotiation stance.

By definition, negotiation is a form of interaction between agents and that is why a negotiation framework requires a language that facilitates such communication (Labrou et al., 1999) [87]. The elements of the communication language are usually referred to as locutions, utterances or speech acts (Searle, 1969; Traum, 1999) [88, 89]. For example, if p is the information conveyed by an utterance or locution or speech act, the information conveyed by the next one can be the acceptance, refusal, challenge, attack, etc. of *p*. Indeed, if agents communicate by exchanging isolated messages, the resulting communication is extremely poor and agents cannot participate in complex interactions such as negotiations, which are formed by a sequence of utterances. Figure 2.2 describes the elements of a classical negotiating agent.

Two major proposals for agent communication languages have been advanced in multi-agent systems, namely the Knowledge Query and Manipulation Language (KQML) (Mayfield et al.,1996) [90] and the Foundation for Intelligent Physical Agents' Agent Communication Language (FIPA ACL) (FIPA, 2001) [91]. For example, FIPA ACL offers 22 locutions. In Chapter 3, we will see how Jadex platform uses FIPA ACL in FIPA Request Interaction Protocol (RP).

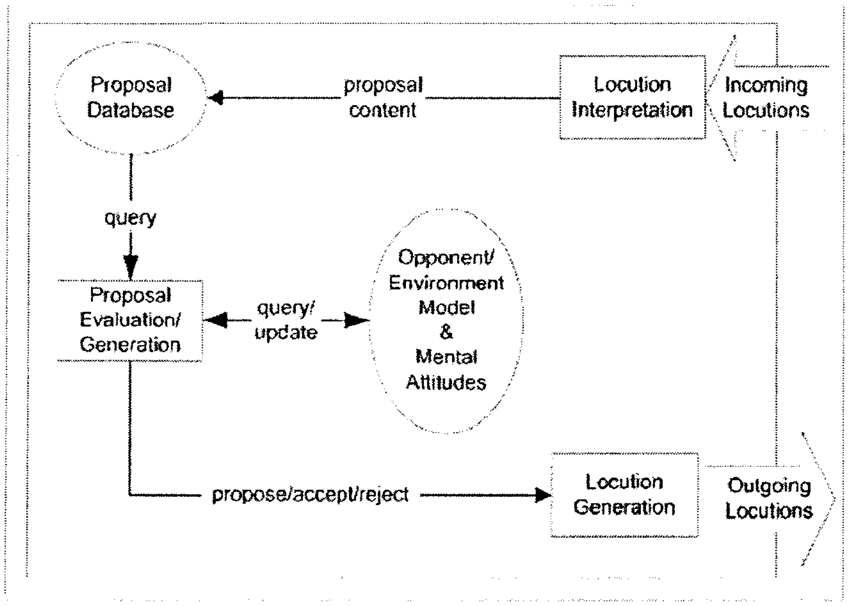


Figure 2.2 Conceptual elements of a classical negotiating agent

In the recent research into agent negotiation, flexible protocols based on dialogue games are used [6, 16, 17]. In Chapter 4, we shall see the important aspects of argumentation in negotiation.

2.4 Trust in Multi-Agent Systems

J. Ousterhout says: "... The agents need to be able to make decisions that are complex and subtle, and we need *to be able to trust them enough that we don't have to check up on them constantly*" [20]. In multi-agent systems, an agent often finds benefit in cooperating with other agents to achieve a payoff, through gaining information or performing actions toward a goal. Cooperation, however in uncertain environments exposes agents to risk. As an example, an agent may believe another agent, which is malicious, and consequently, it may risk its ability to accomplish an intended goal, since the requesting agent cannot be guaranteed that the responding agent will be able to, or will even try to, fulfill the request. In order to evaluate whether to cooperate and ultimately to provide a decision basis for whom to trust, agents must model both the worth and risk of interacting with other agents. Models of trust serve as decision criteria for whether to cooperate with the agent whose trust is being modeled. While explaining the implementation of the proposed trust model in Chapter 5, we will consider other issues related to trust.

2.5 Learning Agents in Multi-Agent Systems

While the agents are referred to be autonomous and intelligent, do not necessarily mean that they are also capable of learning. In our proposals presented in Chapters 4 and 5, all the agents are learning agents (Figure 2.3) which means that they will learn and adapt to changing circumstances. According to Kasabov [22], a learning agent should exhibit the following characteristics:

1. Learn and improve through interaction with the environment (embodiment);

2. Adapt online and in real time;

3. Learn quickly from large amounts of data;

4. Accommodate new problem solving rules incrementally;

5. Have memory based exemplar storage and retrieval capacities;

6. Have parameters to represent short and long term memory, age, forgetting, etc.;

7. Be able to analyze itself in terms of behavior, error and success.

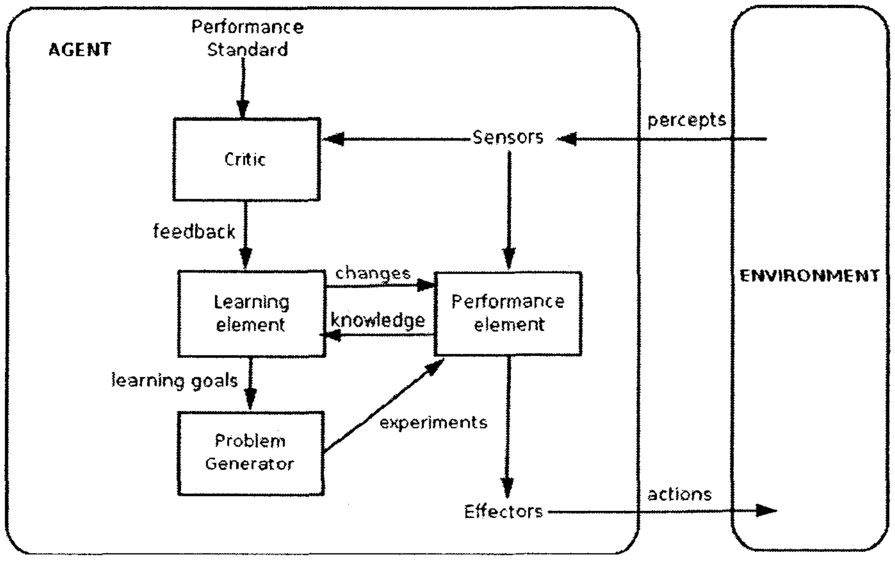


Figure 2.3 Learning Agent

Chapter 3. Multi-Agent Programming

Increasing software and complex systems are using software agents as components to cooperate and coordinate with each other to achieve their expectation. It is becoming more necessary and popular within the domain of agent-based systems that designers need special methodologies to develop software according to the various requirements. Within the last decade, a blooming of agent-based methodologies were introduced and developed based on various theoretical grounds, such as, object-orientation and knowledge representation. Due to lacking of systematic estimation of these approaches, it is difficult to select a proper methodology for designing a particular project. Even only few frameworks were proposed to evaluate those agent-oriented methodologies, however, the measurements in those frameworks are not sound enough. Hence, it is crucial to systematically analyze and evaluate agent-based methodologies to ensure that developers can apprehend what advantages and drawbacks of each methodology are, and which methodology should be chosen when they face several specific complicated systems and projects. The discussed evaluation of methodologies is out of the scope of this thesis work. We have used some of the ideas while designing the implementations mentioned in Chapters 4 and 5. We assessed three prominent agent-oriented methodologies: MaSE, Tropos and Promethus and focused on MaSE. This chapter is organized as follows: Section 3.1 describes two models for agent-based software engineering techniques. Section 3.2 discuses the three agent-oriented methodologies mentioned above. Finally, in Section 3.3 we describe Jadex platform and the features that we have used in our implementations.

3.1 Agent-based Software Engineering Techniques

The most important aspect of the agent-oriented technology is its ability to deal with complexity and emergent behavior of distributed software systems. To construct such complex systems, we need a suitable methodology as a solid foundation to develop the system from the requirement to the implementation stage. Recently, more than two dozen methodologies have been proposed such as: Gaia [23, 30], MaSE [24], MESSAGE [25], Tropos [26], HLIM [27], Prometheus [28], AUML [29], etc. Yet, only recently, the evaluation of these methodologies draws the research community attention. For examples, a comparison among agent-oriented methodologies shows in [25] estimates the similarity between the models of the Gaia [23, 30] and MAS-CommonKADS [31] methodologies. However, it does not explicitly evaluate these methodologies or provide techniques for doing so. A similar comparison is presented in [28], in which the authors contrast between Gaia and MaSE [24] and conclude that MaSE is much more detailed than Gaia. Yet, they do not mention drawbacks of MaSE nor do they provide outlines for making a comparison between methodologies.

In [32], the authors suggest an exemplar case study according to which the various methodologies could be estimated. In addition to the case study, they list a set of questions to be asked about an agent-oriented methodology. However, the questions are somewhat vague and answering these questions may not lead to an understanding of what the right methodology is for a specific project, i.e., there is no framework for evaluating agent-oriented methodologies within that study. Another work on the comparison among agent-oriented methodologies [33] summarizes the advantages and drawbacks of several streams (such as SE, formal methods, and knowledge engineering) within the domain of agent-oriented methodologies. However, it overlooked some of the software engineering aspects of MASs and agent application properties. Additionally, that work did not provide evaluation criteria for assessing advantages and drawbacks of various modeling methods within a specific stream. Before discussing some methodologies, we should introduce some concepts in the following subsections.

3.1.1 BDI-Oriented Model

BDI (Belief, Desire and Intention) is the well-known method to describe rational agents. The motivation of BDI is the recognition that when modeling the behavior of an agent, we should consider the dynamic factors from the system and the environment. BDI describes an agent's beliefs about the system and the environment, the agent's desires (or goals) to achieve as well as expressing the agent's intention by way of executable plans. Agents can reason about what is the best plan for achieving desires under specific beliefs about the environment. An agent can review its goals and respond with revised plans, if necessary, as system or environmental parameters change. Figure 3.1 illustrates these concepts, which convey that intelligent (or cognitive) adaptive systems may comprise three types of processes: reactive, for producing timely responses to external stimuli; deliberative, for possessing learning and reasoning abilities; and reflective, for the ability to continuously monitor and adapt based on introspection. Although useful, the BDI model has limitations for use for the design of multi-agent systems [35, 36].

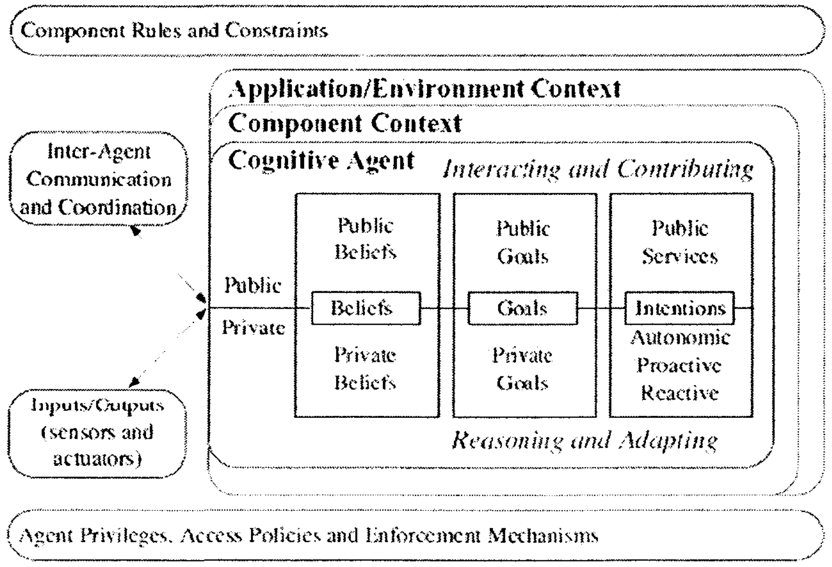

Component Rules and Constraints
Application/Environment Context
Component Context
Cognitive Agent
Interacting and Contributing
Inter-Agent Communication and Coordination
Public Beliefs
Public Goals
Public Services
Public
Private
Beliefs
Goals
Intentions
Autonomic
Proactive
Reactive
Private Beliefs
Private Goals
Inputs/Outputs (sensors and actuators)
Reasoning and Adapting
Agent Privileges, Access Policies and Enforcement Mechanisms

Figure 3.1 Example of BDI model

3.1.2 Role and Society Based Model

We introduce this model from two viewpoints. The first is the social-level point of view and the second is the knowledge-level viewpoint. Figure 3.2 shows the social level model in a summary diagram, which delineate how a system is modeled as an organization or society made up of components, the majority of which are agents. Their communication channels include content and mechanisms, dependencies between agents, and organizational relationships such as the concepts of peers and competitors. In the society, compositional laws are used as guidelines that describe how components in the system are organized under the regulation of the society. Behavioral laws regulate how components (i.e., members in the organization) meet both their roles and societal commitments. From the social level viewpoint, units of the system are different organizations in the society. Different organizational mechanisms and structures can influence the behavior of the constituent components. The way organizational structures change can also significantly affect role relationships, especially by adding/removing roles. The Medium describes how to accomplish these changes, and from the knowledge level side, agents are central to a system. An agent perceives its goals and accomplishes them by actions. These goals and actions are governed by rational rules, which are provided as laws. All laws are based on the knowledge of their environment.

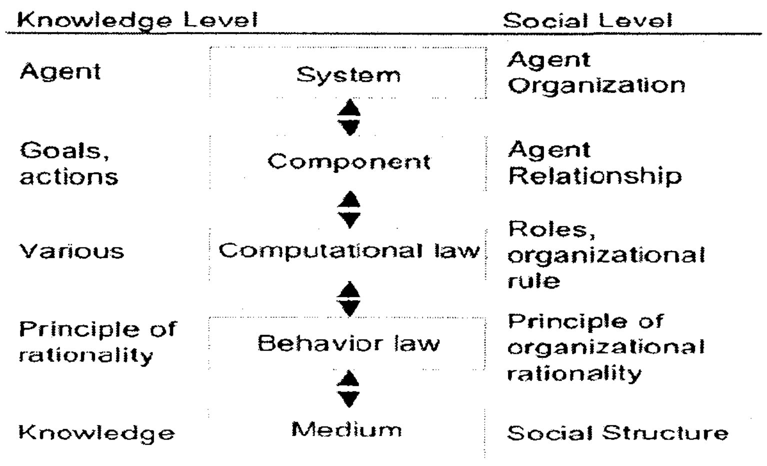


Figure 3.2 Social (knowledge) level model.

3.2 Methodologies

In this section, the three main methodologies of multi-agent software development will be discussed.

3.2.1 Tropos

Tropos [34] was introduced by a research group in University of Toronto, Canada, and is being extended and maintained in some universities in Europe. It was developed for building agent-oriented software systems. Tropos is based on two key ideas. First, the notions of agent and all related mental notions such as goals and plans are used in all phases of software development, from early analysis down to the actual implementation. Second, a crucial role is assigned to requirements analysis and specifications when the system is analyzed with respect to its intended environment. There are five phases in the Tropos design process: (1) early requirements analysis phase: the relevant actors are defined, along with their respective goals; (2) late requirements analysis phase: a potential system actor is introduced and is related to actors in terms of actor dependencies; (3) architectural design phase: more system actors are introduced and they are assigned sub-goals or sub-tasks of the goals and tasks assigned to the system; (4) detailed design phase: system actors are defined in more detail, including specifications of communication and coordination protocols; and (5) implementation phase: the Tropos specifications produced during detailed design phase is transformed into skeleton for the implementation. Tropos adapts JACK programming language for its execution because they are both based on BDI architecture [34, 35].

3.2.2 MaSE

The Multi-agent Systems Engineering (MaSE) is a general-purpose methodology for developing multi-agent systems that is founded on the basis of software engineering principles [24]. MaSE divides the development process into two major phases: the analysis phase and the design phase. For each phase, MaSE provides a set of stages need to be performed. Figure 3.3 presents the development process proposed by MaSE. The analysis phase consists of the following stages: capturing goals, applying use cases and refining roles. The design phase consists of the following stages: creating agent classes, constructing conversations, assembling agent classes and system design.

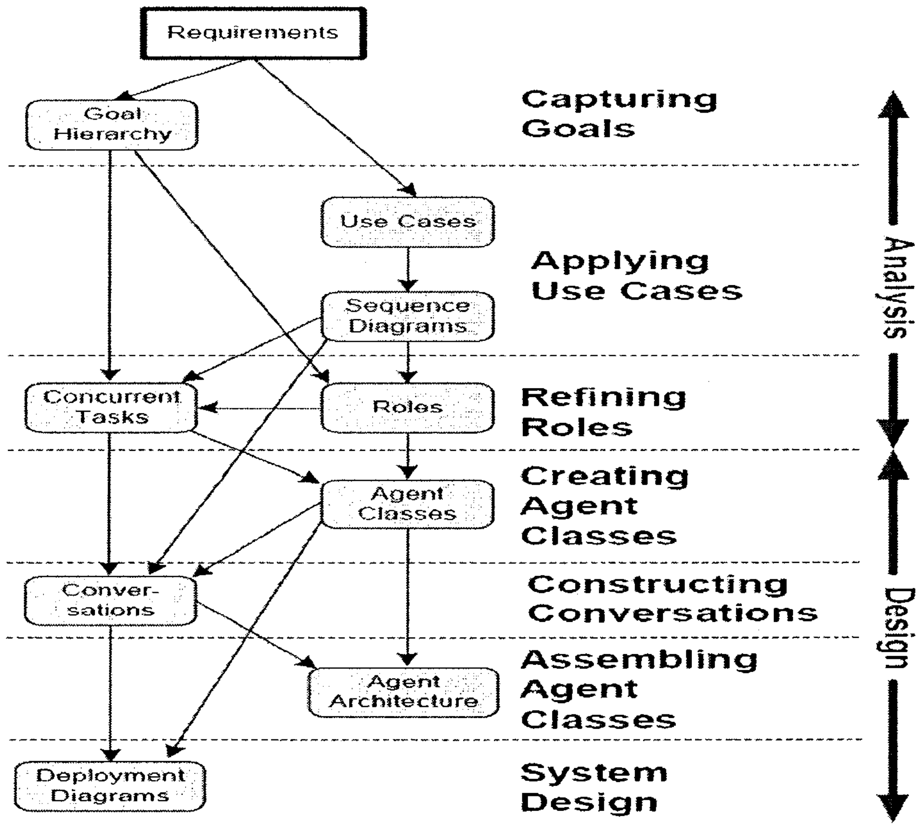


Figure 3.3MaSE Methodology

MaSE methodology deciphers agent-based software design as two main parts: (1) goal analysis, conducted at the beginning of a MaSE process to reinforce goal preservation through analysis and (2) design phases. It facilitates role and agent class modeling to focus on clear goal delegation, where every role is responsible for a particular goal to be accomplished. Every goal has to be associated with a role. With these roles defined, the design of communication between roles and their corresponding tasks becomes fixed, lacking dynamic adaptability of goals.

3.2.3 Prometheus

We consider the overall structure of the Prometheus methodology [28]. Prometheus is intended to be a practical methodology. It aims to be complete: providing everything that is needed to specify and design agent systems.

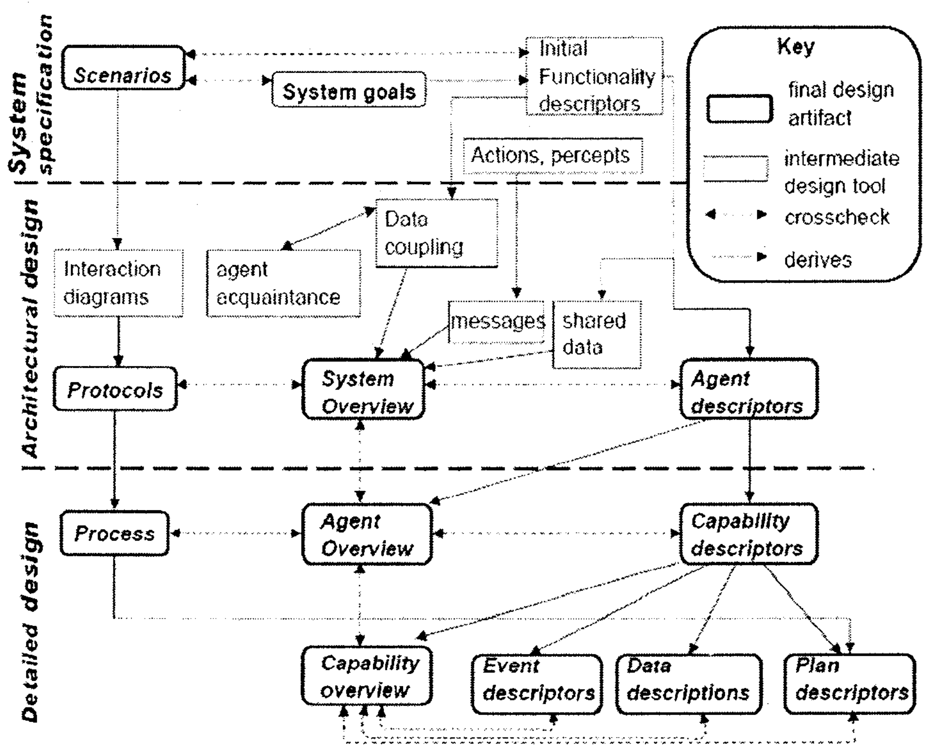


Figure 3.4 Prometheus Methodology

Prometheus methodology consists of three main designing phrases: (1) system specification; (2) architecture design and (3) detailed design. Firstly, system specification begins with a rough idea of the system, which may be simply a few paragraphs of rough description, and proceeds to define the requirements of the system in terms of the goals of the system, use case scenarios, functionalities, and the interface of the system to its environment, defined in terms of actions and percepts.

Several distinguishing features of the Prometheus methodology are below:

1 Prometheus is detailed – it provides detailed guidance on how to perform the various steps that form the process of Prometheus.

2 Prometheus supports the design of agents that are based on goals and plans. A significant part of the benefits that can be gained from agent-oriented software engineering comes from the use of goals and plans to realize agents that are flexible and robust.

3 Prometheus covers a range of activities from requirements specification to detailed design.

4 The methodology is designed to facilitate tool support, and tool support exists in the form of the Prometheus Design Tool (PDT), which is freely available.

Since we have used Jadex as platform, we choose MaSE along with some added and modified techniques for the methodology to design the software, which is the implementation of our negotiation protocol and trust mechanism in Chapters 4 and 5.

3.3 Jadex Platform

Software agent technology is in high demand and many software companies are directing their attention on developing platforms that could be used for the creation of multi-agent environments. Some of these platforms include Jack, Jade, and Jadex. The platform we used in implementing the multi-agent environment is Jadex [37] since it is fully compatible with Belief, Desire and Intention (BDI) model and provides a BDI reasoning engine.

The Jadex system is based on the BDI model and facilitates easy intelligent agent construction with sound software engineering foundations. It uses both XML and Java and can be deployed on different kinds of middleware such as Jade. In order for the creation of agents to happen, agent architecture should take into account agent internal state and artificial intelligence concepts.

The Jadex project [37] accommodates these properties with an open research map that outlines the areas of interest and the actual work in progress in these fields. The framework consists of an API (Application Program Interface), an execution model, and a predefined reusable generic functionality.

The API provides access to the Jadex resources when starting programming plans. Plans are plain Java classes, which could include information such as sending messages, or waiting for events. Jadex has included an intuitive OQL (Object Query Language) used to make queries into databases and information systems. In addition to the plans coded in Java, it provides an XML based Agent Definition File (ADF), which specifies the initial beliefs, goals, and plans of an agent.

In order to develop an agent application in Jadex, one has to create two types of files: XML Agent Definition Files (ADF) (see Appendix 1) and Java classes for the plan implementations. Plans describes the actions that an agent undertakes. The developer needs to define the head and body of the plan. The head contains the conditions in order for the plan to be executed, and these conditions are to be found in the agent definition files. The body is the complete set of steps describing the actions to achieve a goal or reaction. The agent definition file is an XML file that contains the beliefs, goals, and plans of an agent.

3.3.1 Agent Architecture

Our model is implemented on the Jadex platform and as a result, it follows the same agent architecture as the one presented in Figure 3.5. The figure shows how the execution on the agent level takes place in order to produce a message from the plans. The beliefs, goals, plans and events used in this architecture will be described in the following subsections.

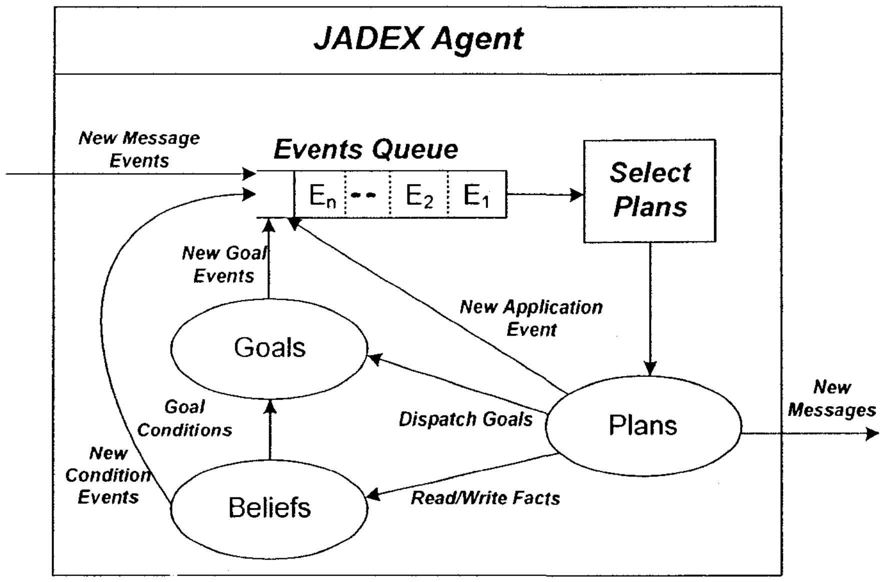


Figure 3.5 Agent Architecture

3.3.2 Beliefbase

A beliefbase is a container that stores believed facts and is an access point for the data in the agent. It provides more abstraction as compared to the attributes in object-oriented world and represents a unified view of the knowledge of an agent. The information about the beliefs, goals, and plans of an agent are included in the ADF. An example of an ADF is shown in Appendix 1. The beliefbase contains strings as the name of a belief that represents an identifier for a specific belief. Since we have two proposed protocols for different aspects in a multi-agent system and we have limitation of space for this thesis, we include only one agent's ADF (Appendix 1) for a consumer agent of the trust model described in Chapter 5. Table 3.1 shows the summary of the mentioned agent's beliefbase [21, 37].

Belief Summary:	
Names	content_of_info Agent has in his knowledge base the names of all the agents present in the framework as a String array.
Circle	agent_list The set of all the agents in the known community as an array list
Mine	agent_categorization The set of agent categorization (agent trust table) as an array list
Known_Tr	Values Trustworthy agent's trust values as a hash table
Known_TR	Values The recency of the information about trustworthy agents as a hash table
Known_N	Values Trustworthy agent's number of interactions as a hash table
Utility	Values The value of the provided service by a service provider agent as an integer

Table 3.1 Beliefbase Summary

3.3.3 Goals

Goals are a central concept in Jadex; they are concrete, momentary desires of an agent. Unlike traditional BDI systems, Jadex treats goals as events. Agents will more or less directly engage into suitable actions, until the goal is being reached. When a goal is adopted, it becomes an option that is added to the agent's desire structure. Some goals may only be valid in specific contexts determined by the agent's beliefs. When the context of a goal is invalid, it will be suspended until the context is valid again. An ADF will include the content of an agent's goal (see Appendix 1) [37]. Table 3.2 shows the content of a goal that represents the agent's desire discussed in the previous section.

Goals Summary:	
Achievegoalref	df_search Search the agents and services in the Directory Facilitator (DF)
Achievegoalref	rp_initiate Initiates the FIPA Request Interaction Protocol (RP) (section 3.3.6)
Maintaingoalref	df_keep_registered Ensures that an agent description remains available at the DF as long as the goal is present in the agent

Table 3.2 Goals Summary

3.3.4 Plans

Plans describe the concrete actions that an agent may carry out to reach its goals. The plan has a head and a body that the developer needs to define. The head contains the conditions under which the plan may be executed and used as specified in the agent definition file (ADF) (appendix 1). The body of the plan, written in JAVA, is a procedural recipe describing the actions to take in order to achieve a goal or react to some event. Table 3.3 shows different plans that agents have in our implementation of the proposed model.

Plans Summary:	
Select	EvaluatonPlan This plan evaluates the trust values as per the proposed protocol and gets services from the providers and updates beliefs if necessary.
Tell	Mine_informPlan This plan informs when asked the other agents of its own trustworthy agents community.
Give	Value_informPlan This plan informs when asked the other agents of its own trustworthy agent's trust value, time relevance and number of interactions it had with that agent.
Final	Utility_informPlan This plan informs when asked about the best service provider agent's rank of the provided service.
Init	InitialPlan The plan initiates the trust table and content table of the agent's in its circle of activity.

Table 3.3 Plans Summary

3.3.5 Events

An important property of agents is the ability to react in a timely fashion to different kinds of events. In Jadex, these events are presented in the ADF program. There exist two types of events, message events and internal events. Internal events can be used to denote an occurrence inside an agent, while message events represent a communication between two or more agents. Events are usually handled by plans [37]. Table 3.4 gives an event summary for our implemented agents for trust management.

Events Summary:	
request_init	direction = "receive" Initiates the initial plan
request_final	direction= "receive" Initiates the utility_inform plan
request_tell	direction= "receive" Initiates the mine_inform plan
request_selection	direction= "receive" Initiates the evaluation plan
request_what	direction= "receive" Initiates the value_inform plan
Inform	direction= "send" This ensures that the important information such as the conversation-id and in-reply-to also appears in the answer. Moreover, message properties, which should not change during a conversation (e.g. protocol, language and ontology) are also automatically copied into the success reply.
Failure	direction= "send" This ensures that the important information such as the conversation-id or in-reply-to also appears in the answer. Moreover, message properties, which should not change during a conversation (e.g. protocol, language and ontology) are also automatically copied into the failure reply.

Table 3.4 Events Summary

3.3.6 Request Interaction Protocol (RP)

The Request Interaction Protocol [37] manages the interaction consisting of one initiator and one participant agent. The initiator wants the participant to perform some action. We have used this protocol through Jadex for our multi-agent system of the both negotiation protocol and trust mechanism proposed in Chapters 4 and 5.

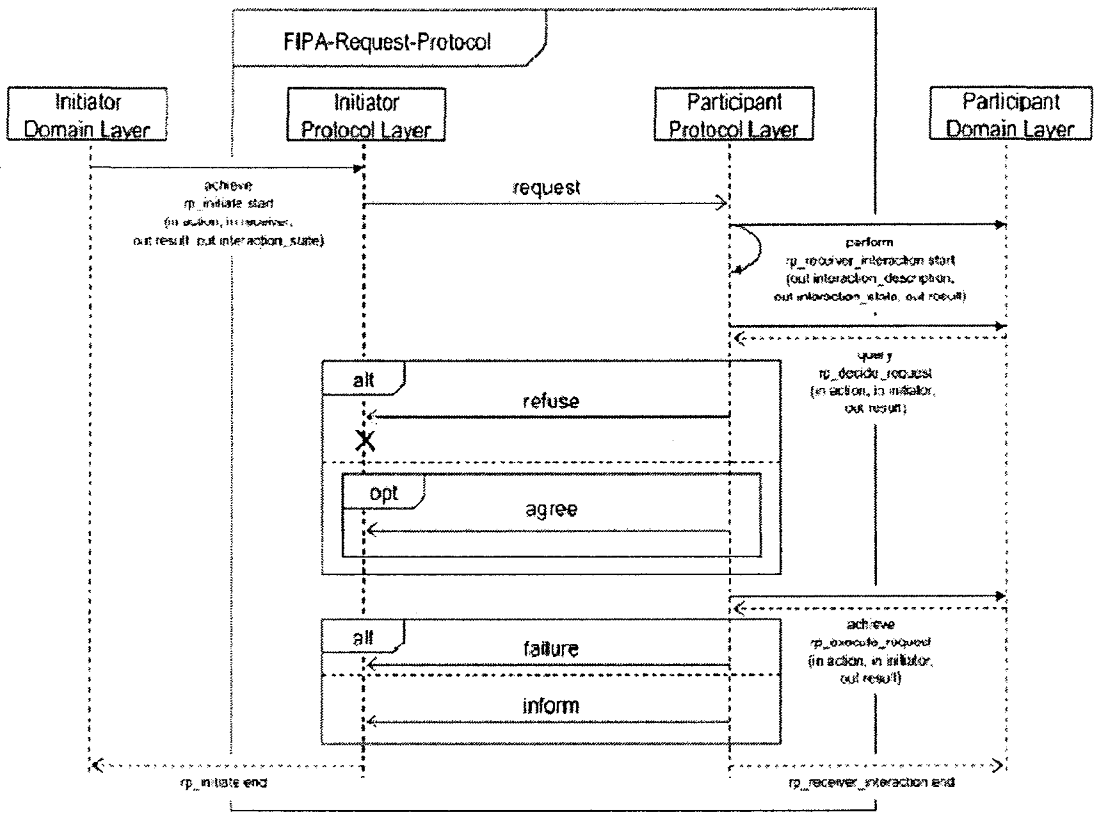


Figure 3.6 The Request Protocol

The protocol consists of an initiator and a participant (Figure 3.6). The initiator asks the participant to perform an action by sending a request message. When the participant receives this message, it accepts or refuses to perform the action, and depending on that decision, it sends either an optional agree message or a refuse message. If it has agreed, the participant subsequently performs the action and when it has finished, it sends a failure or an inform message. The inform message may be just a notification that the task was done or contain a result of the task execution.

3.3.7 Agent's Reasoning Model

After all the discussions about different parameters in agent architecture, it is good to mention here the process of reasoning in Jadex. Jadex facilitates using the BDI model in the context of mainstream programming, by introducing beliefs, goals and plans as first class objects that can be created and manipulated inside the agent. In Jadex, agents have beliefs, which can be any kind of Java object and are stored in a beliefbase. Goals represent the concrete motivations (e.g. states to be achieved) that influence an agent's behavior. To achieve its goals the agent executes plans, which are procedural recipes coded in Java. The abstract architecture of a Jadex agent is depicted in Figure 3.7.

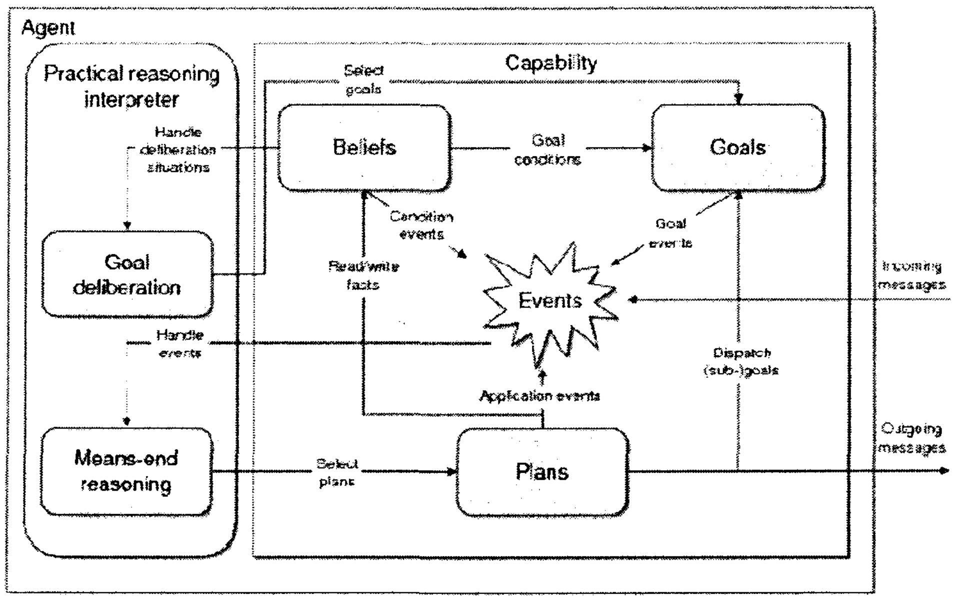


Figure 3.7 Jadex Abstract Architecture

Chapter 4. Designing and Implementing *B2B* Applications Using Argumentative Agents

The design and implementation of *B2B* applications using computational argumentation theory and agent technology is described in this chapter. Section 4.1 introduces the framework from e-business point of view. Section 4.2 describes our agent-based framework for *B2B* applications. Section 4.3 presents the argumentation model upon which this framework operates. Section 4.4 discusses the argumentative protocol for *B2B* conflict resolution and analyzes its formal and computational properties. A case study illustrating this model through a running example is provided in Section 4.5. The implementation of the running example is discussed in Section 4.6.

4.1 Introduction

Our framework for *B2B* applications suggests three levels, *resource*, *application*, and *strategic* that are connected through *rely-on* and *run-on-top-of* relations (Figure 4.1). These levels represent the way businesses generally function: the strategic level, associated with a set of *S*trategic Argumentative Agents (*S*-AAs), sets the goals to reach (e.g., 10% revenue increase). Decisions affecting a business growth are made at this level. The application level, associated with a set of *A*pplication Argumentative Agents (*A*-AAs), sets the automatic and manual processes (e.g., new auditing system) that permit fulfilling these objectives. The resource level, associated with a set of *R*esource Argumentative Agents (*R*-AAs), sets the means that achieve the performance of these processes. The framework couples components (that reside in one of the three levels) with agents equipped with argumentation capabilities to assist a specific component (i) persuade peers of collaborating, (ii) interact with peers during business process implementation, (iii) resolve conflicts that could impede collaboration, and (iv) track conflict resolution. Still in Figure 4.1, rely-on relation means mapping the business objectives onto concrete system applications, and run-on-top-of relation means performing the business processes of these system applications subject to resource availabilities. In addition, both relations make issues at lower levels influence goals at higher levels. For example, lack of resources could result in reviewing goals. In Figure 4.1, horizontal relations permit linking similar levels of separate businesses. We refer to these relations by *interconnectivity*, *composition*, and *collaboration*. Underneath each horizontal relation's name, an example of conflict to fix in a *B2B* scenario is shown for illustration purposes. Collaboration relation bridges the strategic levels and focuses on how businesses adapt their goals and plans so that these businesses can now reach the goals that result out of their decision of partnership. Composition relation bridges the application levels and focuses on how new business processes are developed, either from scratch or after re-engineering existing processes. Finally, interconnectivity relation bridges the resource levels and focuses on the means that make the performance of business processes happen despite distribution and heterogeneity constraints.

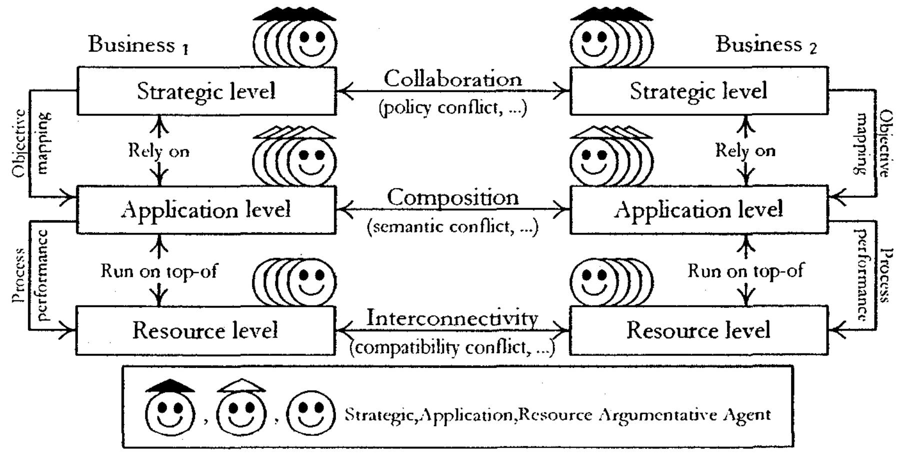


Figure 4.1 The argumentative agent framework for *B2B* applications

4.2 The Proposed Framework for *B2B* Applications

4.2.1 Brief Description of Levels & Relations

The resource level includes data and software resources (e.g., DBMS) that a business owns or manages, and the hardware resources upon which these software resources run.

The application level is about the software applications that businesses operate such as payroll. From a *B2B* perspective, the application level hosts a number of *A*-AAs according to the number of these applications. The role of *A*-AAs is (i) to monitor the external business processes that will make use of software applications and (ii) to initiate interaction sessions with other *A*-AAs. These sessions frame application compositions according to the guidelines that *S*-AAs set and resolve possible conflicts during these compositions as depicted by *composition* relation in Figure 4.1. For illustration purposes, we assume that software applications are implemented as Web services [53], although other technologies could be used.

The strategic level is about the planning and decision-making mechanisms that underpin a business growth. Like the application level, the strategic level hosts a number of *S*-AAs according to the number of active collaborations that a business initiates with its partners. The role of *S*-AAs is (i) to reason over the business plans and (ii) initiate interaction sessions with other *S*-AAs as depicted by *collaboration* relation in Figure 4.1. These sessions aim at persuading peers to participate in collaborations, reviewing policies in case of conflicts, optimizing some parameters such as distribution network, etc. *A*-AAs feed *S*-AAs with details related to the execution progress of business processes. Particularly, if a conflict during the composition process cannot be resolved, *A*-AAs inform their respective *S*-AAs.

From an argumentation perspective, *S*-AAs and *A*-AAs are equipped with the same reasoning capabilities. However, they differ in terms of the knowledge they manage and the responsibilities they are in charge of. For example, to resolve conflicts at the application or strategic levels, *A*-AAs or *S*-AAs use the same persuasion and negotiation protocols but execute them differently. Protocols publicly describe the allowed moves, but how to select a certain move would dependent on the knowledge that feed agents' private strategies.

Figure 4.1 shows vertical and horizontal relations. In a *B2B* context, the focus is on horizontal relations. Interconnectivity relation targets the resource level and allows (i) data to freely and securely flow between businesses without any restriction related to format, location, or semantics and (ii) disparate resources to trigger each other without any restriction related to access rights, time-slot availabilities, or compatibilities. Communication protocol incompatibility (e.g., different vendors) is an example of conflict that falls under interconnectivity relation.

Composition relation targets the application level and exhibits how business processes associated with *A*-AAs get "virtually" integrated without being subject to any functional or structural changes. Lack of common semantics (e.g., different measurement units) is an example of conflict that falls under composition relation. When it comes to Web services-based applications, composition targets users' requests that cannot be satisfied by any single, available Web service, whereas a composite Web service obtained by combining available Web services may be used.

Collaboration relation targets the strategic level and emphasizes the mechanisms that *S*-AAs set-up for coordinating the new *B2B* processes using *A*-AAs. These processes result out of composing applications, stretch beyond businesses' boundaries, and have to consider the requirements/limitations of the resource and application levels per business. Policy incompatibility (e.g., various tax rates) is an example of conflict that falls under collaboration relation. Policies of businesses can be in contradiction, and some core business policies cannot be easily re-engineered. By using argumentative agents, we aim at handling these issues. Through their argumentative reasoning, and interaction, negotiation, and persuasion abilities, these agents could reason about these policies, identify possible conflicts and update their policies to resolve these conflicts. They can also persuade each other for the benefit of collaborating and sharing their resources and determining alternative agents to work with, in case current conflicts cannot be resolved.

4.2.2 Forms of Coordination

We split coordination in the argumentative agent-based framework into two forms: *vertical* between strategic and application levels via rely-on relation, and *horizontal* between strategic or application levels via collaboration or composition relations, respectively. We discuss hereafter how argumentation is used with coordination using Figures 4.2 and 4.3 where plain lines and dotted lines denote interactions and conflict detection/resolution respectively.

Vertical Coordination occurs within the boundaries of the same business. Here an *S*-AA has the authority to execute a set of actions over an *A*-AA (Figure 4.2): “select”, “ping”, “trigger”, and “audit”. These actions are explained as follws:

1. “*select*” action makes the *S*-AA identify the *A*-AA of an application that will pursue the interactions with other *A*-AAs as part of the partnership decision;
2. “*trigger*” action makes the *S*-AA forward the execution requests to the *A*-AA of an application; these requests arrive from others *A*-AAs;
3. “*audit*” action makes the *S*-AA monitor the performance of an application through its *A*-AA; this is needed if the *S*-AA has to guarantee a certain QoS to other *S*-AAs.

Argumentation in vertical coordination is illustrated with two cases: *Application-to-Strategic* (this chapter focus) and *Strategic-to-Application*.

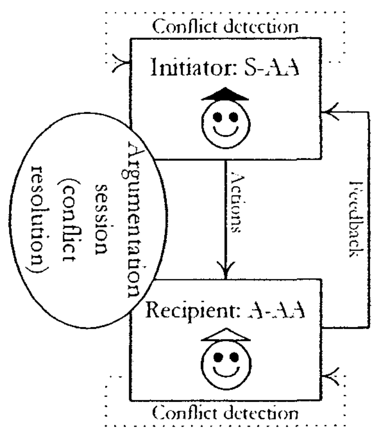


Figure 4.2 Argumentation in vertical coordination

Application-to-Strategic case highlights an *A*-AA that faces difficulties in resolving conflicts and completing its operations. For example, this *A*-AA was put on hold for a long period due to occupied resources or did not receive information in the right format from other businesses' *A*-AAs. As a result, the *A*-AA notifies its *S*-AA so that both set up an argumentation session for the sake of discussing the current difficulties and the potential solutions to put forward. This notification is represented with "feedback" in Figure 4.2. Briefly, we report on the way conflict resolution progresses in this argumentation session.

Case 1. The *S*-AA has an argument supporting the fact that the conflict facing the *A*-AA could be resolved based on similar past situations for example. Thus, the *S*-AA argues with the *A*-AA about the feasibility of this solution using persuasion (Section 4.2.2). If the *A*-AA is not convinced (i.e., persuasion fails), the *S*-AA will decide to select another *A*-AA to continue the uncompleted composition work of the withdrawn *A*-AA.

Case 2. The *S*-AA does not have any argument for or against the possibility of resolving the conflict facing the *A*-AA. Thus, *S*-AA and *A*-AA collaborate to find a solution through an inquiry dialogue game like the one proposed in [42]. As defined byWalton and Krabbe [60], inquiry dialogues rise from an initial situation of general ignorance and the purpose is to achieve the growth of knowledge and agreement.

If neither **case 1** and **case 2** succeed, the respective *S*-AAs of the collaborative businesses try to work out a solution via horizontal coordination. When a solution is found, the *S*-AAs invite the same *A*-AAs if they are still available, or new ones to take part in the composition to deploy at the application level.

Strategic-to-Application case highlights an *S*-AA that expects the occurrence of conflicts if appropriate actions are not taken on time. Examples of actions include reprimanding an *A*-AA that released private details to peers. Expecting conflicts is based on the different feedbacks that the *S*-AA receives from their *A*-AAs. This shows a preventive strategy to conflict occurrence. However, preventive strategies are beyond the scope of this thesis.

Horizontal Coordination spreads over the boundaries of businesses and thus, reflects *B2B* applications in a better way. We identify two scenarios where each scenario involves either *S*-AAs or *A*-AAs. For the sake of simplicity, our description is restricted to *A*-AAs. Here an *A*-AA has the authority to carry out a set of actions over another peer engaged in the same composition (Figure 4.3): "ping" and "trigger".

1. "ping" action makes the *A*-AA check the aliveness of a remote application through its *A*-AA: this is needed before the former *A*-AA submits requests;
2. "trigger" action makes the *A*-AA submit its requests to a remote application through its *A*-AA.

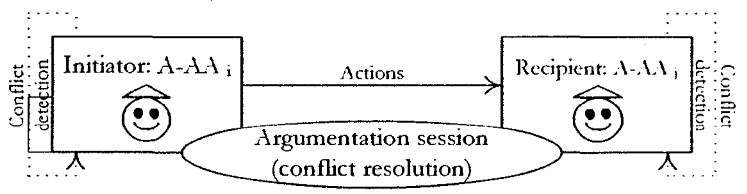


Figure 4.3 Argumentation in horizontal coordination

able to capture properties that other frameworks concentrate on. The set of well-formed formulas (*wff*) built from $\mathcal{L}$ is denoted by $\mathcal{WF}$.

Agents build arguments using their beliefs. The set $Arg(\mathcal{L})$ contains all those arguments. Similarly to [38, 43, 44], we abstractly define argumentation as a dialectical process that underpins the exchange of for/against arguments that lead to some conclusion. Because we are using an abstract language, we are not interested in the internal form of an argument. Formally, we define our argumentation framework as follows:

Definition 1 (Argumentation Framework) *An abstract argumentation framework is a pair* $< A, \mathcal{AT} >$, *where* $A \subseteq Arg(\mathcal{L})$, *and* $\mathcal{AT} \subseteq A \times A$ *is a binary relation over* A *that is not necessarily symmetric. For two arguments* a *and* b, *we use* $\mathcal{AT}(a, b)$ *instead of* $\mathcal{AT} \in \{(a, b)\}$ *to indicate that* a *is an attack against* b.

For example, an argument may be defined as a deduction of a conclusion from a given set of rules, or as a pair (H, h) where h is a sentence in $\mathcal{WF}$ and H a subset of a given knowledge base such that (i) $H \vdash h$, (ii) H is consistent, and (iii) there is no subset of H with properties (i) and (ii).

As conflicts between arguments might occur, we need to define what an acceptable argument is. To this end we define first the notions of "defense" and "admissible set of arguments" (from[44, 45]):

Definition 2 (Defense) *Let* $A \subseteq Arg(\mathcal{L})$ *be a set of arguments over the argumentation framework, and let* $S \subseteq A$. *An argument* a *is defended by* S *iff* $\forall b \in A$ *if* $\mathcal{AT}(b, a)$, *then* $\exists c \in S$: $\mathcal{AT}(c, b)$.

Definition 3 (Admissible Set) *Let* $A \subseteq Arg(\mathcal{L})$ *be a set of arguments over the argumentation framework. A set* $S \subseteq A$ *of arguments is admissible iff:*

1) $\nexists a, b \in S$ *such that* $\mathcal{AT}(a, b)$ *and*

2) $\forall a \in S$ *a is defended by S.*

In other words, a set of arguments is admissible iff it is conflict-free and can counter-attack every attack.

Example 1 Let $\mathrm{A} = \{\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}\}$ and $\mathcal{AT}$ defined as follows: $\mathcal{AT}(\mathrm{b}, \mathrm{a})$, $\mathcal{AT}(\mathrm{c}, \mathrm{a})$, $\mathcal{AT}(\mathrm{d}, \mathrm{b})$, $\mathcal{AT}(\mathrm{d}, \mathrm{c})$. The sets: $\emptyset, \{\mathrm{d}\}$ and $\{\mathrm{a}, \mathrm{d}\}$ are all admissible. However, the sets $\{\mathrm{b}\}$ and $\{\mathrm{d}, \mathrm{c}\}$ are not admissible.

Definition 4 (Characteristic Function) *Let* $A \subseteq Arg(\mathcal{L})$ *be a set of arguments and let S be an admissible set of arguments over the argumentation framework. The characteristic function of the argumentation framework is:*

$$F: 2^A \rightarrow 2^A$$
$$F(S) = \{a | a \text{ is defended by } S\}$$

Definition 5 (Extensions) *Let S be an admissible set of arguments, and let F be the characteristic function of the argumentation framework.*

- *S is a complete extension* (S_{co}) *iff* $S = F(S)$.
- *S is the grounded extension* (S_{gr}) *iff* $S = F(S)$ *and S is minimal (w.r.t. set-inclusion) (grounded extension corresponds to the least fixed point of F).*
- *S is a preferred extension* (S_{pr}) *iff* $S = F(S)$ *and S is maximal (w.r.t. set inclusion).*

Example 2 Let us consider the same argumentation framework as in Example 1.

We have:

- $F(\emptyset) = \{d\}$, so the admissible set $\emptyset$ is not a complete extension.
- $F(\{d\}) = \{a, d\}$, so the admissible set $\{d\}$ is not a complete extension.
- $F(\{a,d\}) = \{a,d\}$, so the admissible set $\{a,d\}$ is a complete extension.

In this example, the only complete extension is $\{a,d\}$ the grounded extension and is also the only preferred extension.

Example 3 Let A={a,b,c} and $\mathcal{AT}$ defined as follows: $\mathcal{AT}$(a, b), $\mathcal{AT}$ (b, a). The sets: {c}, {a, c}, and {b, c} are the complete extensions of the argumentation framework. The minimal complete extension {c} is the grounded extension, and the maximal complete extensions {a, c} and {b, c} are the preferred extensions.

According to Definition 5, an admissible set S is a complete extension if and only if S is a fixed point of the function F, which means that all arguments defended by S are also in S. Also, the grounded extension is the *least fixed point* of F. Consequently, the grounded extension contains all the arguments that are not attacked (the arguments that are defended by the empty set: $F(\emptyset)$), all the arguments that are defended by these non-attacked arguments $F(F(\emptyset)) = F^2(\emptyset)$, all the arguments that are defended by the defended arguments ($F^3(\emptyset)$), and so on until a fixed point is achieved. The grounded extension corresponds to the intersection of all the complete extensions. Finally, a preferred extension is a maximal complete extension that cannot be augmented by adding other arguments while staying complete.

We have the following direct proposition:

Proposition 1 *Let $\langle \mathcal{A}, \mathcal{AT} \rangle$ be an argumentation framework. $\exists! S_{gr}$ in $\langle \mathcal{A}, \mathcal{AT} \rangle$. In words, there exists a single grounded extension for the abstract argumentation framework. Now we can define what the acceptable arguments are in our system.*

Definition 6 (Acceptable Arguments) *Let $A \subseteq Arg(\mathcal{L})$ be a set of arguments, and let $G = S_{gr}$. An argument a over A is acceptable iff $a \in G$.*

Argumentation in horizontal coordination is illustrated with two cases: *Application to-Application* and *Strategic-to-Strategic*.

Application-to-Application case stresses an *A*-AA that identifies a conflict after interacting with peers. Conflicts could be of many types like different security policies, different semantics (e.g. different measurement units, different ontology, etc.), conflicting quality of service, different cost associated with the application, etc.

A-AAs try to resolve these conflicts via argumentation using a combination of *persuasion* and *inquiry* (Section 4.2.2). *A*-AA agents engage in pure persuasion if one of them has already a solution that could be accepted by the other with respect to the beliefs it has. However, merging persuasion with inquiry allows these agents to build up a joint agreed argument.

Strategic-to-Strategic case highlights an *S*-AA that identifies a conflict and tries to resolve it with its *S*-AA partner. Some conflicts at this level concern penalty policies (e.g., collaboration's contract terms and conditions not respected) and payment policies. This case also stresses the situation where two *S*-AAs of the collaborative businesses try to work out a solution of a conflict reported by the respective *A*-AAs which cannot be resolved by vertical coordination. To resolve these conflicts, *S*-AAs engage in persuasions and inquiries (Section 4.2.2). Before presenting this protocol, let us discuss its formal framework based on computational argumentation theory.

4.3 Formal Argumentation System

4.3.1 Generic Background

This section discusses the formal argumentation system that frames the internal operations in our *B2B* framework. This discussion includes the configuration featuring argumentative agents as well. We use an abstract formal language $\mathcal{L}$ to express agents' beliefs. Here abstract means that beliefs could be propositional formulas like in [55], Horn clauses like in [40], or a set of facts and rules like in [42, 47]. The use of an abstract language would make our framework generic and

According to this acceptability semantics, which is based on the grounded extension, if we have two arguments *a* and *b* such that $\mathcal{AT}(a,b)$ and $\mathcal{AT}(b,a)$, then *a* and *b* are both non-acceptable. In a *B2B* scenario, this can happen when two *A*-AAs present two conflicting arguments about the type of security policies to use for the current transaction: a weak policy which is simple to implement and less expensive or a strong policy which is hard to implement and more expensive. This notion is important in *B2B* applications since agents should agree on an acceptable opinion, which is supported by an acceptable argument when a conflict arises. However, during the argumentative conversation, agents could use non-acceptable arguments as an attempt to change the status of some arguments previously uttered by the addressee, from acceptable to non-acceptable. This idea of using non-acceptable arguments in the dispute does not exist in the persuasion and inquiry protocols in the literature. For this reason, we introduce two new types of arguments based on the preferred extensions to capture this notion. We call these arguments *semi-acceptable and preferred semi-acceptable arguments.*

Definition 7 ((Preferred) Semi-Acceptable Arguments) *Let G be the grounded extension in the argumentation framework, and let $E_1,\ldots,E_n$ be the preferred extensions in the same framework. An argument a is:*

- *Semi-acceptable iff $a \not\ni G$ and $\exists\ E_i, E_j$ with $(1 \leq i,\ j \leq n)$ such that $a \in E_i \wedge a \notin E_j$.*
- *Preferred semi-acceptable iff $a \notin G$ and $\forall\ E_i\ (1 \leq i \leq n)\ a \in E_i$.*

In other words, an argument is *semi-acceptable* iff it is not acceptable and belongs to some preferred extensions, but not to all of them. An argument is *preferred semiacceptable* iff it is not acceptable and belongs to all the preferred extensions. Preferred semi-acceptable arguments are stronger than semi-acceptable and grounded arguments are the strongest arguments in this classification.

Example 4 Let A = {a, b, c, d} and $\mathcal{AT}$ is defined as follows: $\mathcal{AT}$ (a, b), $\mathcal{AT}$ (b, a), $\mathcal{AT}$ (a, c), $\mathcal{AT}$ (b, c), $\mathcal{AT}$ (c, d).

- $\emptyset$ is the grounded extension in this argumentation framework.
- The argumentation framework has two preferred extensions: {a, d} and {b, d}. The arguments a and b are then semi-acceptable, and the argument d is preferred semi-acceptable.

A concrete scenario of this example in a *B2B* setting would be as follows: Suppose we have a transaction Tr and three possible security policies for it: s_1, s_2 and s_3. The four arguments a, b, c and d are as follows:

- a: s_1 is the most suitable policy for the transaction Tr.
- b: Alone, s_2 is not sufficient to secure the transaction Tr, but by combining it with s_3 it becomes the most suitable.
- c: s_2 is less expensive than s_1.
- d: s_1 is not expensive to implement, and is sufficient to secure the transaction Tr.

In some extent, the argument d is stronger than a and b because it is defended by these two arguments against the only attacker c, and a and b attacks each other. From a chronological point of view, we can imagine the following scenario leading to build these four arguments at the application level of two businesses represented respectively by A-AA_1 and A-AA_2. First, A-AA_1 presents the argument d, then A-AA_2 attacks by moving forward the argument c. A-AA_1 replies by attacking c using the argument a. At that stage, arguments a and d are grounded. A-AA_2 tries then to degrade one of these two arguments by attacking a using d. A-AA_2 is aware that by using b to attack a, b is at the same time attacked by a. The idea here is just to change the status of the argument presented by A-AA_1 from acceptable to semi acceptable.

Proposition 2 *Let* $A \subseteq Arg(\mathcal{L})$ *be a set of arguments, and let* $SD = \{a \in A \mid \forall\, b \in A\ \ \mathcal{AT}(b, a) \Rightarrow \mathcal{AT}(a, b)\ \&\ \nexists\, c \in A : \mathcal{AT}(c, b)\}$. $\forall a \in SD$, *a is semi acceptable.*

In other words, the arguments defending themselves by only themselves against all the attackers are semi-acceptable.

Proof see Appendix 2.

Proposition 3 *Complete extensions are not closed under intersection.*

Proof see Appendix 2.

Definition 8 (Eliminated Arguments) *Let $A \subseteq Arg(\mathcal{L})$ be a set of arguments, $a \in A$ be an argument, and EL be the set of eliminated arguments over the argumentation framework. Also, let $E_1, \ldots, E_n$ be the preferred extensions in the same framework.*

$a \in EL$ iff $a \notin E_i$, $\forall i \in [1, n]$.

In other words, an argument is eliminated iff it does not belong to any preferred extension in the argumentation framework. We have the following proposition:

Proposition 4 *Let a be an argument in A, and AC, PS, SA be respectively the sets of acceptable, preferred semi-acceptable, and semi-acceptable arguments over the argumentation framework. $a \in EL$ iff $a \notin AC \cup PS \cup SA$.*

In other words, an argument is eliminated iff it is not acceptable, not preferred semi-acceptable, and also not semi-acceptable.

Proof see Appendix2.

Consequently, arguments take four exclusive statuses namely acceptable, preferred semi-acceptable, semi-acceptable, and eliminated. The dynamic nature of agent interactions is reflected by the changes in the statuses of uttered arguments.

4.3.2 Partial Arguments and Conflicts for B2B Applications

In a *B2B* scenario, it happens that argumentative agents *S*-AAs and *A*-AAs do not have complete information on some facts. In similar situation, they can build *partial arguments* for some conclusions out of their beliefs. We define a partial argument as follows:

Definition 9 (Partial Arguments) *Let x be a wff in $\mathcal{WF}$. A partial argument denoted by a_x^p is part of an argument $a \in A$, which misses an argument (or a proof) for x. In other words, by adding a proof supporting x to a_x^p an argument is obtained.*

For example, if arguments are defined as deductions from a set of rules, x will represent some missing rules, and if arguments are defined as pairs (H, h), x will represent a subset of H.

Example 5 let us suppose that arguments are defined as pairs (H, h) using propositional logic. a = ((m,m → n), n) is an argument for n and a_x^p = ((m → n), n) is a partial argument for n missing the support for x = m. a = ((m,m → n, n → l, l → r), r) is an argument for r and a_x^p = ((n → l, l → r), r) is a partial argument for r missing the support for x = n. In this case a possible support is ((m,m → n), n).

In a *B2B* scenario, an example where partial arguments are needed is when *A*- AA$_1$ of business B_1 knows that security policy s_2 that another business B_2 uses can be substituted by policy s_1 that B_1 uses if some conditions are met when deploying s_2. Thus, *A*-AA$_1$ can build a partial argument supporting the fact that B_2 can use s_1. To be an argument, this partial argument needs a support that implementing s_2 in B_2 meets these conditions.

The idea behind building partial arguments by an agent is to check if the other agent can provide the missing part or a part of this missing part so that the complete argument could be jointly built (progressively). This idea which is a part of the inquiry dialogue will be made clear in the persuasion protocol defined in Section 4.4.2.

As for arguments, we need to define what an acceptable partial argument is. This acceptability is defined in the same way as for arguments. We use the notation $a_x^p.x$ to denote the resulting argument of combining the partial argument a_x^p and an argument supporting x supposing that this latter exists.

Definition 10 (Partial Attack) *Let* a_x^p *be a partial argument over the argumentation framework.* $\mathcal{AT}(a_x^p, b)$ *iff* $\mathcal{AT}(a_x^p.x, b)$ *and* $\mathcal{AT}(b, a_x^p)$ *iff* $\mathcal{AT}(b, a_x^p.x)$*.*

Definition 11 (Acceptable Partial Arguments) *A partial argument* a_x^p *is acceptable (preferred semi-acceptable, semi-acceptable) iff* $a_x^p.x$ *is acceptable (preferred semi-acceptable, semi-acceptable).*

Example 6 Let Σ = {n → m, r → l, l → t, ¬ l} be a propositional knowledge base. The partial argument ((n → m),m) is acceptable, however the partial argument ((r → l, l → t), t) is not acceptable since the argument ((r, r → l, l → t), t) is attacked by the argument (¬ l, ¬ l).

***Proposition** 5 Let a be an argument in A. If a is acceptable, then $\forall x \in WF$ a_x^p is acceptable.*

Proof see Appendix 2.

After having specified the argumentation model, We define the notions of conflict and conflict resolution in our *B2B* framework as follows:

***Definition 12 (Conflict)** Let p and q be two wffs in WF. There is a conflict between two argumentative agents α and β about p and q in the B2B framework iff one of them (e.g., α) has an acceptable argument a for p (denoted a $\uparrow$ p) and the other (i.e., β) has an acceptable argument b for q (b $\uparrow$ q) such that $\mathcal{AT}$ (a, b) or $\mathcal{AT}$ (b, a). We denote this conflict by $\alpha_p \not\equiv \beta_q$.*

For example, if p and q represent each a security policy s_1 and s_2 such that s_1 and s_2 cannot be used together, then there is a conflict if one agent has an acceptable argument for using s_1 while the other agent has an acceptable argument for using s_2 (the two arguments are conflicting). This conflict arises when both agents need to agree on which security policy to use.

Before defining the notion of conflict resolution, we need to define the notions of interaction and interaction's outcome. An utterance u made by an agent α in a given interaction is denoted u $\rightsquigarrow$ α.

***Definition 13 (Interaction)** Let α and β be two argumentative agents. An interaction (denoted by $I_{\alpha,\beta}$) between α and β in the B2B framework is an ordering sequence of utterances $u_1, u_2, \ldots, u_n$ such that $u_i \rightsquigarrow \alpha \Rightarrow u_{i-1} \rightsquigarrow \beta$ and $u_i \rightsquigarrow \beta \Rightarrow u_{i-1} \rightsquigarrow \alpha$. CS_α (resp. CS_β) is the set (called commitment store) containing the arguments used by α (resp. β) during the interaction.*

***Definition 14 (Conflict Resolution)** Let p and q be two wffs in WF and α and β be two argumentative agents in the B2B framework such that $\alpha_p \not\equiv \beta_q$. Also let $I_{\alpha,\beta}$ be an interaction in* *this framework. The conflict $\alpha_p \not\equiv \beta_q$ is resolved by the interaction $I_{\alpha,\beta}$ iff the outcome of $I_{\alpha,\beta}$ is a formula $r \in WF$ such that $\exists\ a \in CS_{\alpha}$, $b \in CS_{\beta}$: $a \uparrow r$, $b \uparrow r$ and a and b are both acceptable.*

In the aforementioned security example, the conflict is resolved iff (i) after interaction, one of the agents can build an acceptable argument from its knowledge base and the arguments exchanged during this interaction, supporting the use of the other policy, or (ii) when both agents agree on the use of a new policy such that each agent can build an acceptable argument, from its knowledge base and the exchanged arguments, supporting the use of this policy. The idea here is that by exchanging arguments, new solutions (and arguments supporting these solutions) can emerge. In this case, agents should update their beliefs by withdıawing attacked (i.e. eliminated) assumptions. However, there is still a possibility that each agent keeps its viewpoint at the end of the conversation.

4.4 Argumentative Persuasion for B2B

This section consists of three sub-sections as follows-

4.4.1 Notations

The outcome of an interaction aiming to resolve a conflict in a *B2B* setting depends on the status of the formula representing the conflict topic. As for arguments, a *wff* has four statuses depending on the statuses of the arguments supporting it (an argument supports a formula if this formula is the conclusion of that argument). A *wff* is *acceptable* if there exists an acceptable argument supporting it. If not, and if there exists a preferred semi-acceptable argument supporting it, then the formula is preferred semi-acceptable. Otherwise, the formula is semi-acceptable if a semi-acceptable argument supporting it exists, or eliminated if such an argument does not exist. Let *St* be the set of these statuses. We define the following function that returns the status of a *wff* with respect to a set of arguments:

$\Delta\colon \mathrm{WF} \times 2^{A} \rightarrow \mathrm{St}$

Generally, the interactions we need in a *B2B* scenario involve two argumentative agents. For simplicity, we will not refer in the remainder of the paper to agent types (strategic or application), but denote participating agents by α and β. Each agent has a possibly inconsistent belief base $\Sigma_\alpha$and Σ_β respectively containing, for example, all the policies on which these agents should reason when they manage businesses as explained in previous sections.

Agents use their argumentation systems to decide about the next move to play (e.g., accept or attack the arguments advanced during their interactions). When an agent accepts an argument that an addressee suggests, this agent updates its knowledge base by adding the elements of this argument and removing all the elements that attack this argument. Each agent α has also a commitment store CS_α publicly accessible for reading but only updated by the owner agent. The commitment stores are empty when interaction starts, and updated by adding arguments and partial arguments that the agents exchange. CS_α refers to the commitment store of agent α *at the current moment*.

The possibility for an agent α to build an acceptable argument a (respectively an acceptable partial argument a_x^p) from its knowledge base and the commitment store of the addressee β is denoted by $\mathcal{A}\mathrm{R}(\ \Sigma_\alpha \cup CS_\beta) \rhd a$ (respectively $\mathcal{A}\mathrm{R}(\ \Sigma_\alpha \cup CS_\beta) \rhd a_x^p$). Building a partial argument a_x^p from a knowledge base means that no argument for or against x can be built. $\mathcal{A}\mathrm{R}(\ \Sigma_\alpha \cup CS_\beta) \ntriangleright a$ (respectively $\mathcal{A}\mathrm{R}(\ \Sigma_\alpha \cup CS_\beta) \ntriangleright a_x^p$) means that agent α cannot build an acceptable argument a (respectively an acceptable partial argument a_x^p) from $\Sigma_\alpha \cup CS_\beta$. The symbols $\trianglerighteq$ and $\ntrianglerighteq$ associated with semi-acceptable (partial) arguments are defined in the same way.

4.4.2 Protocol Specification

In our *B2B* framework, agents engage in persuasion and inquiry dialogues to resolve conflicts. Atkinson et al. [39], Pasquier et al. [56], and Prakken [57] propose persuasion protocols for multi-agent systems. However, these protocols consider only pure persuasion without inquiry stages and does not address completeness (or pre-determinism) property [55].We propose a persuasion protocol including inquiry stages for our *B2B* framework, in which pre-determinism is considered. The protocol is modeled with dialogue games [51, 52]. Dialogue games are interactions between players (agents), in which each player moves by performing utterances according to a pre-defined set of rules. Let us define the notions of protocol and dialogue games.

Definition 15 (Protocol) *A protocol is a pair $\langle C, D\rangle$ with C a finite set of allowed moves and D a set of dialogue games.*

The moves in *C* are of *c* different types ($c > 0$).We denote by $M_i(\alpha, \beta, a, t)$ a move of type *i* played by agent α and addressed to agent β at time *t* regarding a content *a*. We consider four types of moves in our protocol: *Assert*, *Accept*, *Attack*, and *Question*. Generally, in the persuasion protocol agents exchange arguments. Except the *Question* move whose content is not an argument, the content of other moves is an argument *a* ($a \in Arg(\mathcal{L})$). When replying to a *Question* move, the content of *Assert* move can also be a partial argument or "?" when the agent does not know the answer. We use another particular move *Stop* with no content. It could be played by an agent to stop the interaction. Intuitively, a dialogue game in *D* is a rule indicating the possible moves that an agent could play following a move done by an addressee. This is specified formally as follows:

Definition 16 (Dialogue Game) *A dialogue game Dg is either of the form:*

$$M_i(\alpha, \beta, a_i, t) \Rightarrow \bigvee_{0<j\leq n_i} M_j(\beta, \alpha, a_j, t')$$

where M_i, M_j are in $\mathcal{C}$, $t < t'$ and n_i is the number of allowed moves that β could perform after receiving a move of type i from α;

or of the form:

$$\Rightarrow \bigvee_{0<j\leq n} M_j(\alpha, \beta, a_j, t_0)$$

where M_j is in $\mathcal{C}$, t_0 is some initial time, and n is the number of allowed moves that α could perform initially.

According to this definition, a dialogue game is in general non-deterministic, in that, for example, given an incoming move of type i, the receiving agent needs to choose amongst n_i possible replies. As proposed in [40, 41, 42], we combine public dialogue games with private strategies so that agents become deterministic. To this end we introduce the conditions within dialogue games, each associated with a single reply.

Definition 17 (Strategic Dialogue Game) *A strategic dialogue game SDg is a conjunction of rules, specified either as follows:*

$$\bigwedge_{0<j\leq n_i} (M_i(\alpha, \beta, a, t) \wedge C_j \Rightarrow M_j(\beta, \alpha, a_j, t'))$$

where $t < t'$ and n_i is the number of allowed communicative acts that β could perform after receiving a move of type i from α;

or as follows:

$$\bigwedge_{0<j\leq n} (C_j \Rightarrow M_j(\alpha, \beta, a_j, t_0))$$

where t_0 is the initial time and n is the number of allowed moves that α could play initially.

In order to guarantee determinism, conditions C_j need to be mutually exclusive [58]. Agents use their argumentation systems to evaluate, in a private manner, conditions C_j. These argumentation systems are based on the private agents' beliefs and the public commitments recorded in the commitment stores.

To simplify the notations, we omit the time parameter form the moves and use the notation $\cup CS$ as an abbreviation of $CS_\alpha \cup CS_\beta$. In our $\mathcal{B}$usiness-to-$\mathcal{B}$usiness $\mathcal{P}$ersuasive $\mathcal{P}$rotocol ($\mathcal{B2B}$-$\mathcal{PP}$), agents are not allowed to play the same move (with the same content) more than one time. The strategic dialogue games we consider in this protocol are:

1- Initial game

$$C_{in1} \Rightarrow Assert(\alpha, \beta, a)$$

where:

$$C_{in1} = \exists p, q \in \mathcal{WF}: \alpha_p \not\equiv \beta_q \wedge \mathcal{AR}(\Sigma_\alpha) \rhd a \wedge a \uparrow p$$

The persuasion starts when a conflict is detected and one of the two agents asserts an acceptable argument supporting its position. In the remainder of this section, we suppose that the persuasion topic is represented by the *wff* p.

2- Assertion game

$Assert(\alpha, \beta, \mu) \wedge C_{as1} \Rightarrow Attack(\beta, \alpha, b)\ \wedge$

$Assert(\alpha, \beta, v) \wedge C_{as2} \Rightarrow Question(\beta, \alpha, x) \wedge$

$Assert(\alpha, \beta, v) \wedge C_{as3} \Rightarrow Accept(\beta, \alpha, a) \wedge$

$Assert(\alpha, \beta, v) \wedge C_{as4} \Rightarrow Stop(\beta, \alpha)$

where μ is an argument or partial argument, v is an argument, partial argument, or "?" and:

$$C_{as1} = Op_{as_1}^{at_1} \vee (\neg Op_{as_1}^{at_1} \wedge Op_{as_1}^{at_2})$$

$$Op_{as_1}^{at_1} = \exists b \in A{:}\, \mathcal{AR}(\Sigma_\beta \cup CS_\alpha) \rhd b$$

$$\wedge\, \Delta(p, \cup\, CS) \neq \Delta(p, \cup\, CS \cup \{b\})$$

$$Op_{as_1}^{at_2} = \exists b \in A{:}\, \mathcal{AR}(\Sigma_\beta \cup CS_\alpha) \unrhd b$$

$$\wedge\, \Delta(p, \cup\, CS) \neq \Delta(p, \cup\, CS \cup \{b\})$$

$$C_{as2} = \neg C_{as1} \wedge (Op_{as_2}^{qu_1} \vee (\neg Op_{as_2}^{qu_1} \wedge Op_{as_2}^{qu_2}))$$

$$Op_{as_2}^{qu_1} = \exists b_x^p, b_x^p.x \in A{:}\, \mathcal{AR}(\Sigma_\beta \cup CS_\alpha) \rhd b_x^p$$

$$\wedge\, \Delta(p, \cup\, CS) \neq \Delta(p, \cup\, CS \cup \{b_x^p.x\})$$

$$Op_{as_2}^{qu_2} = \exists b_x^p, b_x^p.x \in A{:}\, \mathcal{AR}(\Sigma_\beta \cup CS_\alpha) \unrhd b_x^p$$

$$\wedge\, \Delta(p, \cup\, CS) \neq \Delta(p, \cup\, CS \cup \{b_x^p.x\})$$

$$C_{as3} = \exists a \in A{:}\, \mathcal{AR}(\Sigma_\beta \cup CS_\alpha) \rhd a \wedge a \uparrow p$$

$$\wedge\, \neg Op_{as_2}^{qu_1} \wedge \neg Op_{as_2}^{qu_2}$$

$$C_{as4} = \neg Op_{as_1}^{at_1} \wedge \neg Op_{as_2}^{qu_1} \wedge \neg Op_{as_2}^{qu_2} \wedge \neg C_{as3}$$

$$\wedge\, \forall b \in A, \mathcal{AR}(\Sigma_\beta \cup CS_\alpha) \unrhd b \Rightarrow$$

$$\Delta(p, \cup\, CS) = \Delta(p, \cup\, CS \cup \{b\})$$

In this game, the content of $Assert$ could be an argument, partial argument, or "?". Indeed agents can use this move to assert new arguments in the initial game or to reply to a question in the question game, which is a part of *inquiry* in our protocol. The move that agent β can play as a reply to the $Assert$ move depends on the content of this assertion. When α asserts an argument or a partial argument, CS_α gets changed by adding the advanced (partial) argument. Agent β can attack agent α if β can generate an acceptable argument from its knowledge base and the α's commitment store so that this argument will change the status of the persuasion topic. Consequently, in this protocol agents do not attack only the last advanced argument, but any advanced argument during the interaction, which is still acceptable or (preferred) semi-acceptable

($Op_{as_1}^{at_1}$). This makes the protocol more flexible and efficient (for example agents can try different arguments to attack a given argument). If such an acceptable argument cannot be generated, β will try to generate a (preferred) semi-acceptable argument changing the status of p ($Op_{as_1}^{at_2}$). The idea here is that if β cannot make α's arguments eliminated, it will try to make them (preferred) semi-acceptable. This is due to the following proposition whose proof is straightforward from the definition of semi-acceptable arguments and the fact that only four statuses are possible.

Proposition 6 *If β plays the Attack move with a semi-acceptable argument, then the α's attacked argument changes the status from acceptable to semi-acceptable, and the persuasion topic changes the status from acceptable to semi-acceptable or preferred semi-acceptable.*

We notice that in Assertion game changing the status of p is a result of an attack relation:

Proposition 7 *In Assertion game we have:* $\forall b \in A$,

$\Delta(p, \cup CS) \neq \Delta(p, \cup CS \cup \{b\}) \Rightarrow \exists a \in \cup CS: \mathcal{AT}(b, a)$.

If β cannot play the *Attack* move, then before checking the acceptance of an α's argument, it checks if no acceptable and then no (preferred) semi-acceptable argument in the union of the knowledge bases can attack this argument (inquiry part). For that, if β can generate a partial argument changing the status of p, then it will question α about the missing assumptions ($Op_{as_2}^{qu_1}$ and $Op_{as_2}^{qu_2}$). This new feature provides a solution to the "pre-determinism" problem identified in [55]. If such a partial argument does not exist, and if β can generate an acceptable argument supporting p, then it plays the *Accept* move (C_{as3}).

Proposition 8 *An agent plays the Accept move only if it cannot play the Attack move and cannot play the Question move.*

Proof see Appendix 2.

Agent β plays the $Stop$ move when it cannot accept an α's argument and cannot attack it. This happens when an agent has a semi-acceptable argument for p and the other a semi-acceptable argument against p, so the status of p in the union of the commitment stores will not change by advancing the β's argument (C_{as4}). Finally, we notice that if the content of $Assert$ move is "?", β cannot play the $Attack$ move. The reason is that such an $Assert$ is played after a question in the Question game, and agents play $Question$ moves only if an attack is not possible. By simple logical calculus, we can prove the following proposition:

Proposition 9 *An agent plays the* $Stop$ *move iff it cannot play another move.*

3- Attack game

$$Attack(\alpha,\beta,a) \wedge C_{at1} \Rightarrow Attack(\beta,\alpha,b) \quad \wedge$$

$$Attack(\alpha,\beta,a) \wedge C_{at2} \Rightarrow Question(\beta,\alpha,x) \quad \wedge$$

$$Attack(\alpha,\beta,a) \wedge C_{at3} \Rightarrow Accept(\beta,\alpha,a) \quad \wedge$$

$$Attack(\alpha,\beta,a) \wedge C_{at4} \Rightarrow Stop(\beta,\alpha)$$

Where:

$$C_{at1} = Op_{at_1}^{at_1} \vee (\neg Op_{at_1}^{at_1} \wedge Op_{at_1}^{at_2})$$

$$Op_{at_1}^{at_1} = Op_{as_1}^{at_1}$$

$$Op_{at_1}^{at_2} = Op_{as_1}^{at_2}$$

$$C_{at2} = \neg C_{at1} \wedge (Op_{at_2}^{qu_1} \vee (\neg Op_{at_2}^{qu_1} \wedge Op_{at_2}^{qu_2}))$$

$$Op_{at_2}^{qu_1} = Op_{as_2}^{qu_1}$$

$$Op_{at_2}^{qu_2} = Op_{as_2}^{qu_2}$$

$$C_{at3} = \mathcal{AR}(\Sigma_\beta \cup CS_\alpha) \rhd a \wedge \neg Op_{at_2}^{qu_1} \wedge \neg Op_{at_2}^{qu_2}$$

$$C_{at4} = \neg Op_{at_1}^{at_1} \wedge \neg Op_{at_2}^{qu_1} \wedge \neg Op_{at_2}^{qu_2} \wedge \neg C_{at3}$$

$$\wedge \forall b \in A, \mathcal{AR}(\Sigma_\beta \cup CS_\alpha) \unrhd b \Rightarrow$$

$$\Delta(p, \cup CS) = \Delta(p, \cup CS \cup \{b\})$$

The conditions associated with the Attack game are similar to the ones defining the Assert game. The *Attack* move also includes the case where the agent that initiates the persuasion puts forward a new argument, which is not attacking any existing argument but changing the status of the persuasion topic. This is useful when the advanced arguments cannot be attacked/defended, so that the agent tries another way to convince the addressee.

4- Question game

$$Question(\alpha, \beta, x) \wedge C_{qu1} \Rightarrow Assert(\beta, \alpha, a) \qquad \wedge$$

$$Question(\alpha, \beta, x) \wedge C_{qu2} \Rightarrow Assert(\beta, \alpha, y_{x'}^{p}) \qquad \wedge$$

$$Question(\alpha, \beta, x) \wedge C_{qu3} \Rightarrow Assert(\beta, \alpha, ?)$$

Where:

$$C_{qu1} = \exists a \in A: \mathcal{AR}(\Sigma_\beta \cup CS_\alpha) \rhd a \wedge (a \upharpoonleft x \vee a \upharpoonleft \bar{x})$$

$$C_{qu2} = \exists y_{x'}^{p}, y_{x'}^{p}. x' \in A: \mathcal{AR}(\Sigma_\beta \cup CS_\alpha) \rhd y_{x'}^{p}$$

$$\wedge (y_{x'}^{p} \upharpoonleft x \vee y_{x'}^{p} \upharpoonleft \bar{x})$$

$C_{qu3} \quad = \neg C_{qu1} \wedge \neg C_{qu2}$

Agent β can answer the α's question about the content x by asserting an argument for or against x. If not, it answers by a partial argument if it can generate it. Otherwise, it answers by "?" which means that it does not know if x holds or not. We recall that this game is played when an agent has a partial argument and asks the addressee about the missing part, so that the answer could be the complete missing part, a part of it, or nothing.

5- Stop game

$$Stop(\alpha,\beta) \wedge C_{st1} \Rightarrow Question(\beta,\alpha,x) \qquad \wedge$$

$$Stop(\alpha,\beta) \wedge C_{st2} \Rightarrow Stop(\beta,\alpha)$$

Where:

$C_{st1} \quad = Op_{st_1}^{qu_1} \vee (\neg Op_{st_1}^{qu_1} \wedge Op_{st_1}^{qu_2}))$

$Op_{st_1}^{qu_1} \quad = Op_{as_2}^{qu_1}$

$Op_{st_1}^{qu_2} \quad = Op_{as_2}^{qu_2}$

$C_{st2} \quad = \neg C_{st1}$

Before answering the α's *Stop* move by another *Stop* to terminate the persuasion, β checks if no other partial arguments changing the status of p could be generated. Consequently, the *Stop* move is played only if no such argument could be generated, which means that the conflict cannot be resolved.

4.4.3 Protocol Analysis

In this section, we prove the termination, soundness, and completeness of $\mathcal{B2B}$-$\mathcal{PP}$.

Theorem 1 *$\mathcal{B}2\mathcal{B}$-$\mathcal{PP}$ always terminates either successfully by Accept or unsuccessfully by Stop.*

Proof see Appendix 2.

When the protocol terminates, we define its soundness and completeness as follows:

Definition - Completeness 18 (Soundness-Completeness) *A persuasion protocol about a wff p is sound and complete iff for some arguments a for or against p we have:*

$\mathcal{AR}(\Sigma_\alpha \cup \Sigma_\beta) \rhd a \Leftrightarrow \mathcal{AR}(\cup\, CS) \rhd a.$

Theorem 2 *The protocol ($\mathcal{B}2\mathcal{B}$ -$\mathcal{PP}$) is sound and complete.*

Proof see Appendix 2.

4.5 Case Study

Our running example illustrates a purchase-order scenario (Figure 4.4). A customer places an order for products via Customer-WS (WS for Web service). Based on this order, Customer-WS obtains details on the customer's purchase history from CRM-WS (Customer Relationship Management) of Business $\mathcal{B}_1$. Afterward, Customer-WS forwards these details to $\mathcal{B}_1$'s Billing-WS, which calculates the customer's bill based on these details (e.g., considering if the customer is eligible for discounts) and sends the bill to CRM-WS. This latter prepares the detailed purchase order based on the bill and sends Inv-Mgmt-WS (Inventory Management) of $\mathcal{B}_1$ this order for fulfillment. For those products that are in stock, Inv-Mgmt-WS sends Shipper-WS of $\mathcal{B}_2$ a shipment request. Shipper-WS is now in charge of delivering the products to the customer. For those not in stock, Inv-Mgmt-WS sends Supplier-WS of $\mathcal{B}_3$ a supply message to the requisite, which provides the products to Shipper-WS for subsequent shipment to the customer.

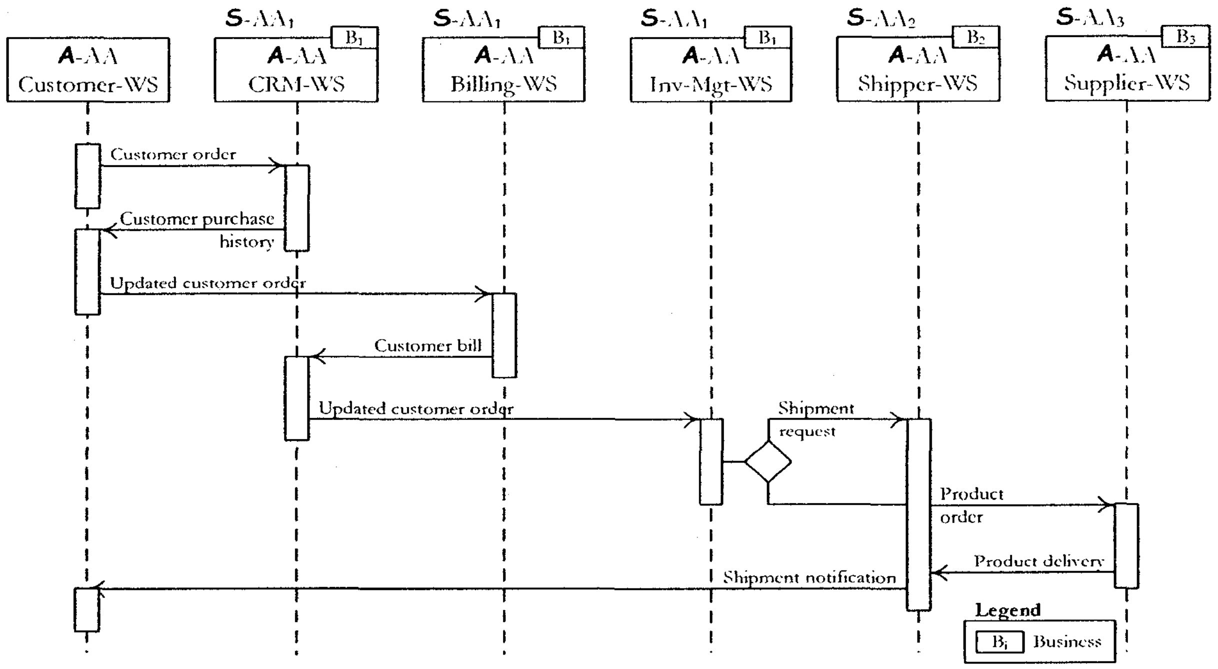


Figure 4.4 Specification of purchase-order scenario

The above scenario could be affected by several types of conflicts. For example, $\mathcal{B}_2$'s Shipper-WS may not deliver the products as agreed with $\mathcal{B}_1$'s Inv-Mgmt-WS, perhaps due to lack of trucks. This is an application-level conflict that needs to be resolved using our $\mathcal{B2B}$-$\mathcal{PP}$ by which, Shipper-WS tries to persuade Inv-Mgmt-WS about the new shipment time and then inform Customer-WS of the new delivery time. If not, Shipper-WS may change its policies by canceling its partnership agreements without prior notice. This is a strategic-level conflict, that calls for either asking $\mathcal{B}_2$ to which Shipper-WS belongs to review its policies, or if that does not work, selecting an alternate shipper.

Let α_{B_1} be the A-AA of Inv-Mgmt-WS and β_{B_2} be the A-AA of Shipper-WS. The resolution of the application level conflict along with the use of dialogue games are hereafter provided:

1- β_{B_2} identifies the conflict and plays the Initial game by asserting an acceptable argument a about lack of trucks from its $\Sigma_{\beta_{B_2}}$ supporting its position: $Assert(\beta_{B_2}, \alpha_{B_1}, a)$.

2- α_{B_1} has an argument b attacking β_{B_2}'s argument which is about available trucks committed to others that could be used to ship the products. α_{B_1} plays then the Assertion game by advancing the $Attack$ move: $Attack(\alpha_{B_1}, \beta_{B_2}, b)$.

3- β_{B_2} replies by playing the Attack game. Because it does not have an argument to change the status of the persuasion topic, but has a partial argument for that, which is about the high price of these particular trucks that could be not accepted by α_{B_1}, it advances the move: $Question(\beta_{B_2}, \alpha_{B_1}, x)$ where x represents accepting or not the new prices. The idea here is that β_{B_2} can attack α_{B_1}, if it refuses the new prices that others have accepted.

4- α_{B_1} plays the Question game and answers the question by asserting an argument c in favor of the increased shipment charges: $Assert(\alpha_{B_1}, \beta_{B_2}, c)$.

5- β_{B_2} plays the Assertion game, and from $\Sigma_{\beta_{B_2}} \cup CS_{\alpha_{B_1}}$, it accepts the argument and agrees to deliver the products as per the agreed schedule with the new price, which is represented by d: $Accept(\beta_{B_2}, \alpha_{B_1}, d)$. Consequently, the persuasion terminates successfully by resolving the conflict.

Figures 4.5 and 4.6 illustrate the scenario details with the exchanged arguments.

1- Conflict detection by A-AA of Shipper-WS after the request of the product P1 with normal delivery time from A-AA of Inv-Mgmt-WS. There is a conflict because A-AA of Inv-Mgmt-WS has an acceptable argument from its knowledge base for normal delivery time which is:

$p, p \rightarrow q$

where

p = "past agreement", and q = "normal delivery time of P1"

And A-AA of Shipper-WS has an acceptable argument $"a"$ for delayed delivery of P1 which is:

$a = r, r \rightarrow s$

Where r = "luck of trucks to ship product P1", and s = "delayed delivery of P1"

Here q and s are contradictory, hence the conflict. The formula s represents the conflict topic.

A-AA of Shipper-WS plays the Initial game by asserting his acceptable argument $"a"$ about lack of trucks

2- A-AA of Inv-Mgmt-WS has an argument $"b"$ in its knowledge base attacking A-AA of Shipper-WS's argument which is:

$b = t_1, t_2, t_1 \wedge t_2 \rightarrow u$

where t_1 = "some trucks *tr* committed to another businesses *BS*", t_2 = "trucks *tr* could be used to ship the product P1", and u = "available trucks to ship product P1"

Here u and r are contradictory. Inv-Mgmt Agent plays then the Assertion game by advancing the Attack move with the argument $"b"$.

3- At this stage, A-AA of Shipper-WS cannot change the state of the conflict topic by attacking the Inv-Mgmt Agent's argument. However, it has a partial argument for that, which is about the high price of these particular trucks that could be not accepted by Inv-Mgmt Agent. The partial argument is:

$m, m \wedge x \rightarrow r$

where m = "price of trucks *tr* is *pr*", x = "Inv-Mgmt Agent's not accept price *pr*", and r = "luck of trucks to ship product P1". This is a partial argument because it needs x to be an argument. For that, A-AA of Shipper-WS plays the Attack game with the *Question* move about x.

4- Inv-Mgmt Agent's has an argument from its knowledge base against x. It plays the Question game and answers the *Question* move by asserting an argument $"c"$ in favor of the increased shipment charges. This argument is:

$k, k \rightarrow l$

Where k = "*pr* is less than Max", and l = "accept *pr*"

(l and x are contradictory)

5- From the Inv-Mgmt Agent's commitment store and the A-AA of Shipper-WS's knowledge base, this latter plays an *Accept* move in which it accepts to deliver the product P1 with normal delivery time and the new price *pr*.

Figure 4.5 Scenario description

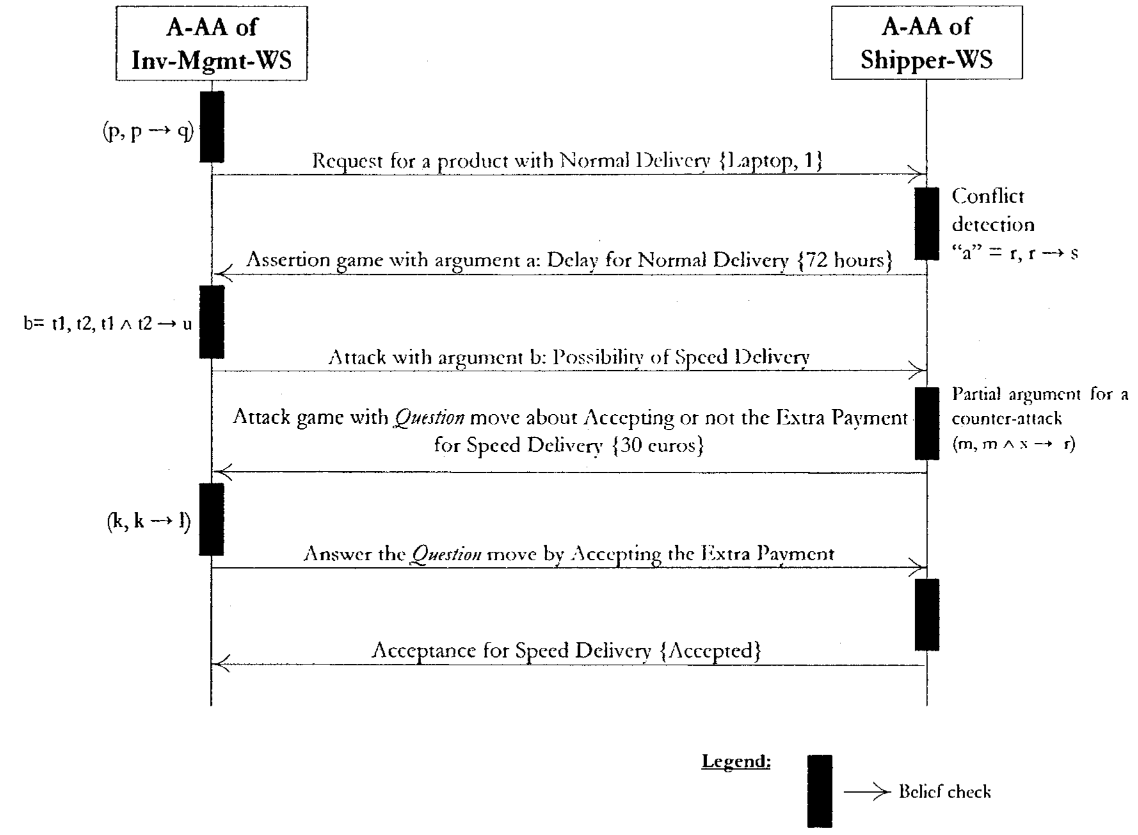

A-AA of Inv-Mgmt-WS
A-AA of Shipper-WS
(p, p → q)
Request for a product with Normal Delivery {Laptop, 1}
Conflict detection "a" = r, r → s
Assertion game with argument a: Delay for Normal Delivery {72 hours}
b= t1, t2, t1 ∧ t2 → u
Attack with argument b: Possibility of Speed Delivery
Partial argument for a counter-attack (m, m ∧ s → r)
Attack game with Question move about Accepting or not the Extra Payment for Speed Delivery {30 euros}
(k, k → l)
Answer the Question move by Accepting the Extra Payment
Acceptance for Speed Delivery {Accepted}
Legend:
Belief check

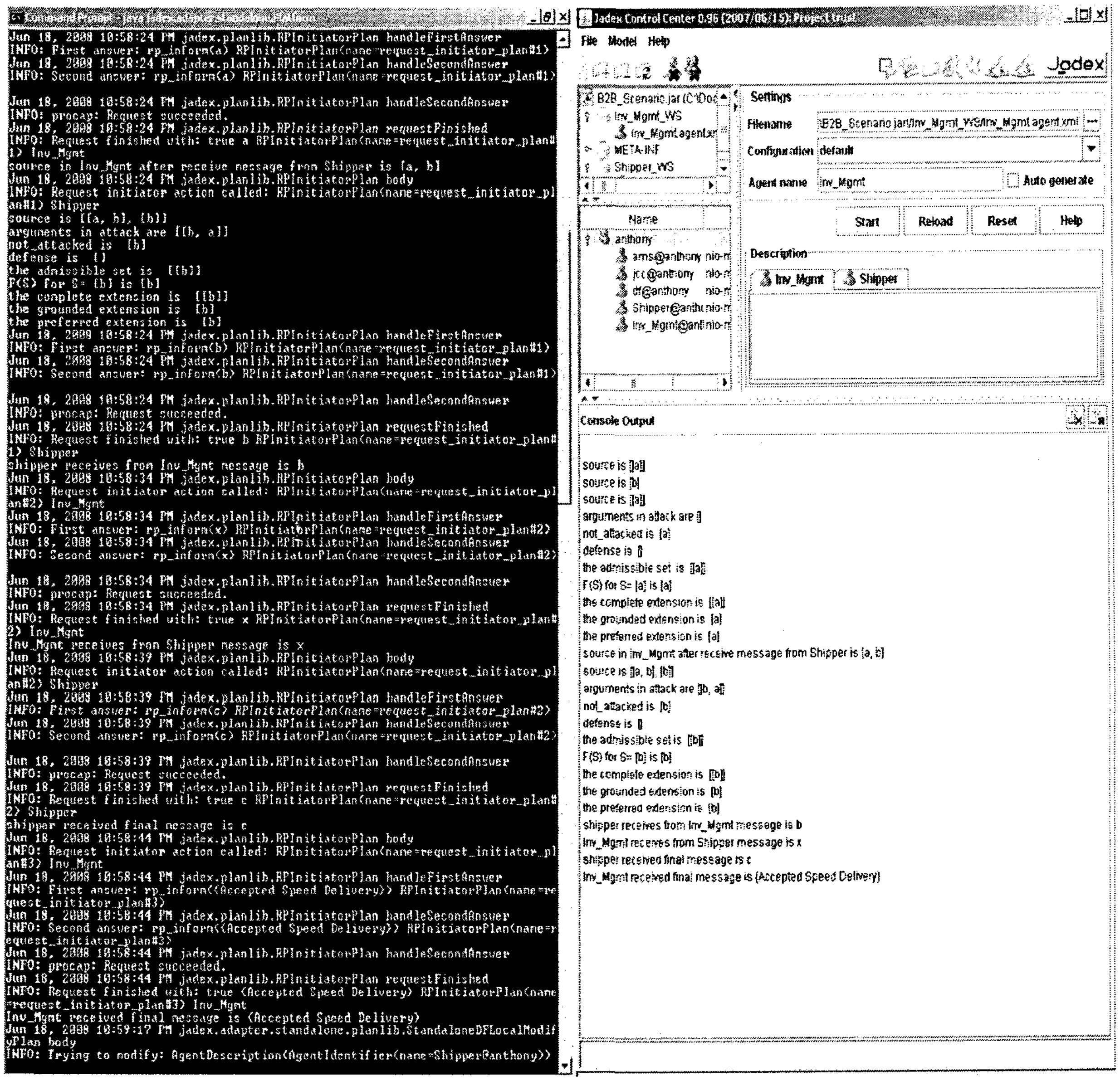
Figure 4.6 Sample of interaction between *A*-AA of Inv-Mgmt-WS and *A*-AA of Shipper-WS

4.6 Implementation

The main challenge of implementing the protocol above is to find the grounded and preferred extensions dynamically from an argumentation framework as discussed in Section 4.1. We used the word dynamic because the argumentation framework will change in each interaction between agents. Only few works have been done in terms of implementation to solve this challenge. In CaSAPI [62] the argumentation system is developed on Prolog and it only tests if an argument belongs to some extensions, which means that whether the argument holds or not. Another implemented system called Java argumentation tool kit "ArgKit" [63] is developed recently, which also performs the same functionality (testing if an argument holds).

Because we have chosen Jadex platform, which is fully based on java (Section 3.3), a java based implementation for the argumentation framework is necessary. It is possible to have prolog based implementation of such framework. However, combining java and prolog has still some problems even though there are some projects facilitating this process for example 'tuProlog' [65] and 'JLog' [64]. Our implementation is the first one that can generate grounded and preferred extensions (if any) from an argumentation framework.

Using this implementation for argumentation framework, we have implemented a proof-of-concept prototype of the scenario discussed in the previous section (Figures 4.5, 4.6) using the Jadex Agent System. Both agents have the same java class to find *acceptable*, *(preferred) semi-acceptable* arguments from grounded or preferred extensions. Figure 4.7 depicts a screenshot of the prototype illustrating the computation of the arguments in the scenario. Currently, the prototype only demonstrates the case of horizontal coordination described above. Future extensions would include other scenarios as well as vertical coordination.



```
Jun 18, 2008 10:58:24 PM jadex.planlib.RPInitiatorPlan handleFirstAnswer
INFO: First answer: rp_inform(a) RPInitiatorPlan(name=request_initiator_plan#1)
Jun 18, 2008 10:58:24 PM jadex.planlib.RPInitiatorPlan handleSecondAnswer
INFO: Second answer: rp_inform(a) RPInitiatorPlan(name=request_initiator_plan#1)

Jun 18, 2008 10:58:24 PM jadex.planlib.RPInitiatorPlan handleSecondAnswer
INFO: procap: Request succeeded.
Jun 18, 2008 10:58:24 PM jadex.planlib.RPInitiatorPlan requestFinished
INFO: Request finished with: true a RPInitiatorPlan(name=request_initiator_plan#
1) Inv_Mgmt
source in Inv_Mgmt after receive message from Shipper is [a, b]
Jun 18, 2008 10:58:24 PM jadex.planlib.RPInitiatorPlan body
INFO: Request initiator action called: RPInitiatorPlan(name=request_initiator_pl
an#1) Shipper
source is [[a, b], [b]]
arguments in attack are [[b, a]]
not_attacked is  [b]
defense is  []
the admissible set is  [[b]]
F(S) for S= [b] is [b]
the complete extension is  [[b]]
the grounded extension is  [b]
the preferred extension is  [b]
Jun 18, 2008 10:58:24 PM jadex.planlib.RPInitiatorPlan handleFirstAnswer
INFO: First answer: rp_inform(b) RPInitiatorPlan(name=request_initiator_plan#1)
Jun 18, 2008 10:58:24 PM jadex.planlib.RPInitiatorPlan handleSecondAnswer
INFO: Second answer: rp_inform(b) RPInitiatorPlan(name=request_initiator_plan#1)

Jun 18, 2008 10:58:24 PM jadex.planlib.RPInitiatorPlan handleSecondAnswer
INFO: procap: Request succeeded.
Jun 18, 2008 10:58:24 PM jadex.planlib.RPInitiatorPlan requestFinished
INFO: Request finished with: true b RPInitiatorPlan(name=request_initiator_plan#
1) Shipper
shipper receives from Inv_Mgmt message is b
Jun 18, 2008 10:58:34 PM jadex.planlib.RPInitiatorPlan body
INFO: Request initiator action called: RPInitiatorPlan(name=request_initiator_pl
an#2) Inv_Mgmt
Jun 18, 2008 10:58:34 PM jadex.planlib.RPInitiatorPlan handleFirstAnswer
INFO: First answer: rp_inform(x) RPInitiatorPlan(name=request_initiator_plan#2)
Jun 18, 2008 10:58:34 PM jadex.planlib.RPInitiatorPlan handleSecondAnswer
INFO: Second answer: rp_inform(x) RPInitiatorPlan(name=request_initiator_plan#2)

Jun 18, 2008 10:58:34 PM jadex.planlib.RPInitiatorPlan handleSecondAnswer
INFO: procap: Request succeeded.
Jun 18, 2008 10:58:34 PM jadex.planlib.RPInitiatorPlan requestFinished
INFO: Request finished with: true x RPInitiatorPlan(name=request_initiator_plan#
2) Inv_Mgmt
Inv_Mgmt receives from Shipper message is x
Jun 18, 2008 10:58:39 PM jadex.planlib.RPInitiatorPlan body
INFO: Request initiator action called: RPInitiatorPlan(name=request_initiator_pl
an#2) Shipper
Jun 18, 2008 10:58:39 PM jadex.planlib.RPInitiatorPlan handleFirstAnswer
INFO: First answer: rp_inform(c) RPInitiatorPlan(name=request_initiator_plan#2)
Jun 18, 2008 10:58:39 PM jadex.planlib.RPInitiatorPlan handleSecondAnswer
INFO: Second answer: rp_inform(c) RPInitiatorPlan(name=request_initiator_plan#2)

Jun 18, 2008 10:58:39 PM jadex.planlib.RPInitiatorPlan handleSecondAnswer
INFO: procap: Request succeeded.
Jun 18, 2008 10:58:39 PM jadex.planlib.RPInitiatorPlan requestFinished
INFO: Request finished with: true c RPInitiatorPlan(name=request_initiator_plan#
2) Shipper
shipper received final message is c
Jun 18, 2008 10:58:44 PM jadex.planlib.RPInitiatorPlan body
INFO: Request initiator action called: RPInitiatorPlan(name=request_initiator_pl
an#3) Inv_Mgmt
Jun 18, 2008 10:58:44 PM jadex.planlib.RPInitiatorPlan handleFirstAnswer
INFO: First answer: rp_inform(<Accepted Speed Delivery>) RPInitiatorPlan(name=re
quest_initiator_plan#3)
Jun 18, 2008 10:58:44 PM jadex.planlib.RPInitiatorPlan handleSecondAnswer
INFO: Second answer: rp_inform(<Accepted Speed Delivery>) RPInitiatorPlan(name=r
equest_initiator_plan#3)
Jun 18, 2008 10:58:44 PM jadex.planlib.RPInitiatorPlan handleSecondAnswer
INFO: procap: Request succeeded.
Jun 18, 2008 10:58:44 PM jadex.planlib.RPInitiatorPlan requestFinished
INFO: Request finished with: true <Accepted Speed Delivery> RPInitiatorPlan(name
=request_initiator_plan#3) Inv_Mgmt
Inv_Mgmt received final message is <Accepted Speed Delivery>
Jun 18, 2008 10:59:17 PM jadex.adapter.standalone.planlib.StandaloneDFLocalModif
yPlan body
INFO: Trying to modify: AgentDescription(AgentIdentifier(name=Shipper@anthony))
```

Filename B2B_Scenario.jar/Inv_Mgmt_WS/Inv_Mgmt.agent.xml
Configuration default
Agent name Inv_Mgmt
Auto generate

```
source is [[a]]
source is [b]
source is [[a]]
arguments in attack are []
not_attacked is  [a]
defense is  []
the admissible set is  [[a]]
F(S) for S= [a] is [a]
the complete extension is  [[a]]
the grounded extension is  [a]
the preferred extension is  [a]
source in Inv_Mgmt after receive message from Shipper is [a, b]
source is [[a, b], [b]]
arguments in attack are [[b, a]]
not_attacked is  [b]
defense is  []
the admissible set is  [[b]]
F(S) for S= [b] is [b]
the complete extension is  [[b]]
the grounded extension is  [b]
the preferred extension is  [b]
shipper receives from Inv_Mgmt message is b
Inv_Mgmt receives from Shipper message is x
shipper received final message is c
Inv_Mgmt received final message is (Accepted Speed Delivery)
```

Figure 4.7 A screenshot from the prototype -computing arguments-

4.7 Related Work

Recent years have seen a continuing surge of interest in designing and deploying *B2B* applications. Service-oriented architecture is the most widely methodology that have been used in this field [50, 54, 59, 61]. In [50] the author proposes the exploitation of Web services and intelligent agent techniques for the design and development of a *B2B* e-commerce application. A multi-party multi-issue negotiation mechanism is developed for this application. This negotiation is a Pareto optimal negotiation based on game theory. This proposal aims at achieving an agreement by computing concessions and generating offers in order to maximize the utility of the participating agents. However, unlike our argumentation-based framework, this mechanism cannot be used to resolve general conflicts as those discussed in Sections 4.2 and 4.3. In [54], the authors develop a methodology for *B2B* design applications using Web-based data integration. The aim is the creation of adaptable semantics oriented meta-models to facilitate the design of mediators by considering several characteristics of interoperable information systems such as extensibility and composability. The methodology is used to build cooperative environments involving the integration of Web data and services. Unlike our methodology, this proposal does not consider conflicts that can arise during the cooperation phase and only addresses the cooperation from technological point of view.

On the other side, and from an argumentation viewpoint, some interesting protocols for persuasion and inquiry have been proposed. [39] propose Persuasive Argument for Multiple Agents (PARMA) Protocol, which enables participants to propose, attack, and defend an action or course of actions. This protocol is specified using logical consequence and denotational semantics. The focus of this work is more on the semantics of the protocol rather than the dynamics of interactions. [42] propose a dialogue-game inquiry protocol that allows two agents to share knowledge in order to construct an argument for a specific claim. There are many fundamental differences between this protocol and ours. Inquiry and persuasion settings are completely different since the objectives and dynamics of the two dialogues are different. In [42], argumentation is captured only by the notion of argument with no attack relation between arguments. This is because agents collaborate to establish joint proofs. However, in our system, agents can reason about conflicting assumptions, and they should compute different acceptability semantics, not only to win the dispute, but also to reason internally in order to remove inconsistencies from their assumptions. From the specification perspective, there are no similarities between the two protocols. Our protocol is specified as a set of rules about which agents can reason using argumentation, which captures the agents' choices and strategies. However, in [42] the protocol is specified in a declarative manner and the strategy is only defined as a function without specifying how the agents can use it. The adopted moves in the two proposals are also different. Another technical, but fundamental difference in the two protocols is the possibility in our protocol of considering not only the last uttered argument, but any previous argument which allows agents to consider and try different ways of attacking each other.

Chapter 5. Trust Management in Open Multi-Agent Systems [1]

In the previous chapter, we discussed how *B2B* applications are designed using open multi-agent systems, where participating agents communicate by exchanging messages through a communication protocol. In a *B2B* setting, businesses through their representative agents are distributed in large-scale network and mutually interact to coordinate and share services with other agents. In such open multi-agent systems, agents should trust each other before starting their collaboration activities and establishing partnerships. The purpose of this chapter is to present a trust framework for these open agent-based systems. For simulation purposes, we assume that we have a set of businesses providing services (service providers) and a set of customers. Some of service providers are trustworthy and some are malicious. When a customer selects a service provider, he obtains some utilities depending on the trust level of the provider. The trust mechanism is evaluated in terms of the gained utility by agents using this mechanism. The mechanism should allow agents to identify and then select the best service providers.

This chapter is organized as follows. In Section 5.1, we discuss the background of the trust mechanism. Section 5.2 describes the agent structure we are using in our trust framework. The trust evaluation technique is detailed in Section 5.3. Trust computing with maintenance, based on the fact that agents in our model have learning capabilities, is shown in Section 5.4. The implementation in different environments is described in Section 5.5. Finally, a discussion of the advantages of the proposed model is presented in Section 5.6.

1 *This chapter is essentially derived from the following papers:*

a. Khosravifar, Babak; Bentahar, Jamal; Alam, Rafiul and Gomrokchi, Maziar; CRM: Comprehensive Reputation Model for Open Multi-Agent Systems; submitted at IEEE Transactions on Knowledge and Data Engineering (**TKDE**).

b. Khosravifar, Babak; Bentahar, Jamal; Gomrokchi, Maziar and Alam, Rafiul; An approach to Comprehensive Trust Management in Multi-Agent Systems with Credibility; IEEE 3rd International Conference on Research Challenges in Information Sciences, RCIS, Marrakech, Morocco 2009.

5.1 Background

Agent's trust in another is the measure of willingness that the agent will make what it agrees to do [66, 67, 68]. Attempting to maintain a trust-based approach, different frameworks have been proposed representing the trust agents have in one another. The most recent research works in trust models are as follows: a) interaction trust, which is based on the direct interactions among involved parties; b) witness reputation, which is based on the reports provided by the third parties; and c) certified reputation, which is based on the references requested from some agents to report their beliefs about a particular agent's behavior.

The proposed frameworks objectively emphasize collecting the involved features in the trust assessments. The objective is to collect reliable information, which leads to an accurate trust assessment procedure. However, since agents are self-interested, there is always the possibility of gaining fake information by a particular agent, even considering the certified reputation provided by the *target agent* (the agent to be evaluated) [69]. In this case, the final trust rate would be affected with non-reliable information about the target agent and eventually the evaluator's imagination about the target agent will not be true. These frameworks generally do not act properly as agents in dynamic environment tend to change their goals and consequently their behaviors. Moreover, these models do not recognize the recent improvement or degradation in particular agent's capabilities.

Generally, trust models using direct experience need long term of interaction to reach a state that agents can evaluate trust level of each other [41, 70, 71]. This is done either by direct experience used to estimate the trust level of these agents or by moving to the second level of evaluation process, asking other agents that are known to be trustworthy about the credibility of the target agents. However, there is a problem if such *trustworthy agents* are not able to report on these agents. Moreover, the trust is not a transitive relationship (the fact that agent A is trustworthy according to agent B and agent B is trustworthy according to agent C, does not mean that agent A is also trustworthy according to agent C). We aim at overcoming these limitations by proposing a framework combining the use of *trustworthy agents* (introduced by the evaluator agent) and *referee agents* (introduced by the target agent) and by considering the possible changes in agents' behaviors.

Another contribution of our trust mechanism is that the *requesting agents* (i.e. the agents requesting information about a target agent) perform maintenance after a period of direct interaction with a new agent in order to adjust the trustworthiness of the consulting agents who provided information regarding to the trust level of the new agent. In the maintenance process, the suggestions provided by other agents are compared with the actual behavior of the new agent in direct interaction. Exceeding some predefined thresholds, the evaluator agent would either increase or decrease its belief about the consulting agent. Doing so, gradually more accurate ratings about the other agents would be dispersed around the environment. This allows us to obtain a better trust assessment. The characteristics and efficiency of our model, called *CRM* (*Comprehensive Reputation Model*) are presented in the next sections.

5.2 Agent Architecture

In our framework, agents are equipped with Beliefs, Desires and Intentions (BDI) (see Section 3.1.1). They use the BDI architecture when they interact with each other. Establishing the trust between two agents is the quite frequent event that agents carry on, as they are involved in interactions. Either evaluating or being evaluated, a rational agent is following strategies to make the best decisions. Figure 5.1 illustrates the overall agent structure.

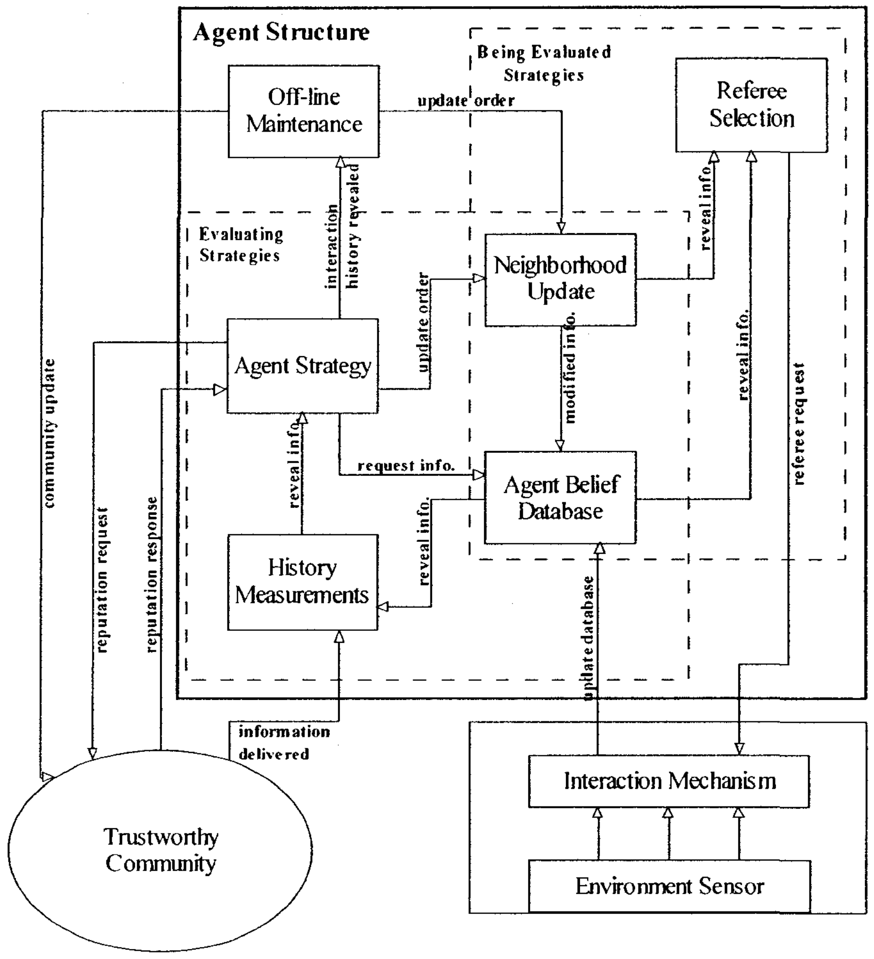


Figure 5.1 Agent structure equipped with BDI architecture

Suppose that an agent Ag_a wants to evaluate the trustworthiness of a target agent Ag_b with who he never (or not enough) interacted before. Ag_a may want to consult some other agents to get better and more accurate information about Ag_b's reputation. In this process, there are two types of interfering agents, the ones known by Ag_a, which are called *trustworthy agents* and the ones known by Ag_b, which are called *referee agents*. The *referee agents* are introduced by Ag_b to report on his trust level based on the past experience (Figure 5.2).

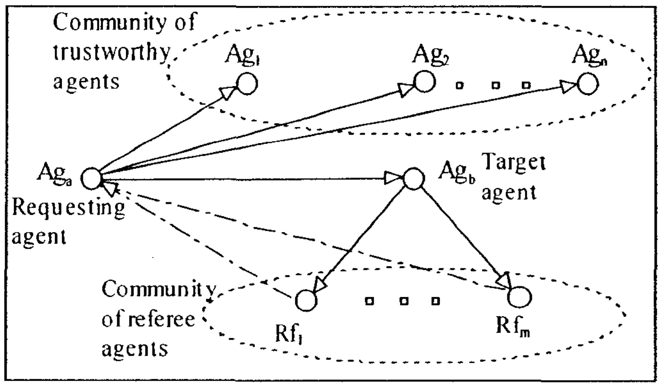


Figure 5.2 Trustworthy and referee agent's topology

Therefore, from agent structure point of view, each agent has its own trustworthy community containing the agents who are the most reliable for him. This community would be known as the agents that are being asked for information in case the agent is evaluating some other agent or being asked for providing recommendation in case the agent is being evaluated by some other agent (Figure 5.3). In this figure, Req_Inf stands for request for trust related information, Req_Ref stands for request for references, Ask_Ref stands for ask for information from the introduced referees, Rep_Inf stands for reply by providing the requested information, Rep_Refuse stands for replying by refusing to provide the requested information and finally Rep_NotHave stands for replying by informing the request or that the requested information is not available.

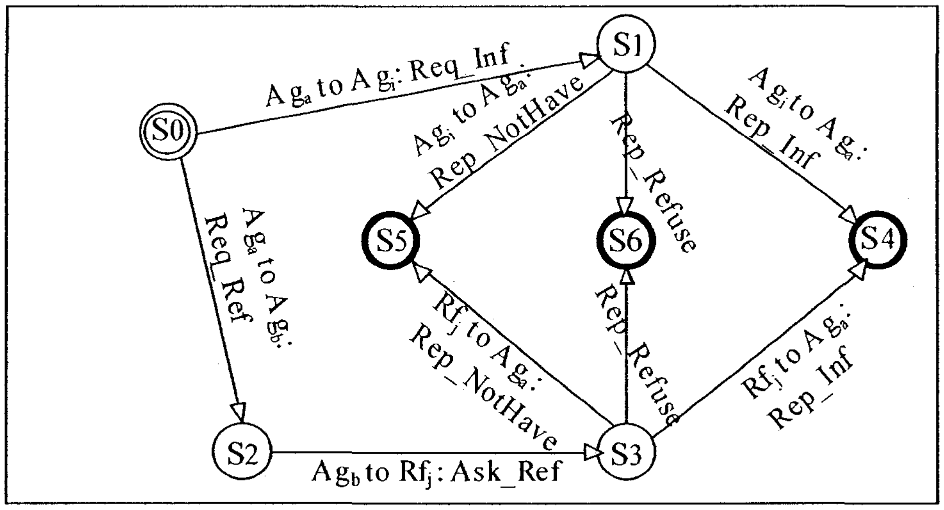


Figure 5.3 Protocol of gathering information from trustworthy and referee agents

Each agent has a strategy component (Figure 5.1), which performs the main evaluation process. The agent requests the history measurements of the previous direct interaction or it refers to the agent belief database for its belief about others. In dynamic systems, the beliefs are always subject to change, and this causes modification in each individual agent's trustworthy community or neighborhood. Therefore, we use in our agent structure a component, which makes the updates in the neighborhood with which the agent is interacting.

5.3 Trust Computing with Maintenance

Let *A* be a set of agents, and *D* be a set of domains or topics. The trust function *Tr* associates two agents from *A* and a domain from *D* with a trust value between 0 and 1:

$$Tr : A \times A \times D \rightarrow [0,1]$$

Given some concrete agents Ag_a and Ag_b in *A* and some concrete domain *D*, $Tr(Ag_a, Ag_b, D)$ stands for "the trust value associated to the target agent Ag_b in domain *D* by the requesting agent Ag_a".

It is obvious that judging based on the accumulated ratings would represent unfairness, as all the interactions would be treated equally and factors like time and the value of the transactions are not considered. Therefore, some trust metrics are to be taken into account to adjust the confidence to some certain extent. To simplify the notation, in the remainder we will omit the domain from all the formulas. Given agents Ag_a and Ag_b in A, we will represent $Tr(Ag_a, Ag_b)$ in short as $Tr_{Ag_a}^{Ag_b}$.

Let Δt be the time difference between the current time and the time at which requesting agent updates its information about the target agent's trust. Equation 1 gives an estimation of $Tr_{Ag_a}^{Ag_b}$.

$$Tr_{Ag_a}^{Ag_b}(\Delta t) = \frac{\Omega(Ag,n,Ag_a,Ag_b)+\Psi(Rf,m,Ag_a,Ag_b)}{\Omega\prime(Ag,n,Ag_a,Ag_b)+\Psi\prime(Rf,m,Ag_a,Ag_b)} \qquad (1)$$

where:

$$\Omega(Ag,n,Ag_a,Ag_b) = \sum_{i=1}^{n} Tr_{Ag_a}^{Ag_i} \times Tr_{Ag_i}^{Ag_b} \times N_{Ag_b}^{Ag_i}$$

$$\Omega'(Ag,n,Ag_a,Ag_b) = \sum_{i=1}^{n} Tr_{Ag_a}^{Ag_i} \times N_{Ag_b}^{Ag_i}$$

$$\Psi(Rf,m,Ag_a,Ag_b) = \sum_{j=1}^{m} Tr_{Ag_a}^{Rf_j} \times Tr_{Rf_j}^{Ag_b} \times N_{Ag_b}^{Rf_j}$$

$$\Psi'(Rf,m,Ag_a,Ag_b) = \sum_{j=1}^{m} Tr_{Ag_a}^{Rf_j} \times N_{Ag_b}^{Rf_j}$$

This equation is composed of two different terms representing the values got from two different consulting communities involved in trust evaluation. The function Ω is defined as the summation of the trust values estimated by the *trustworthy agents* together with their related self-trustworthiness and the number of interactions between the *trustworthy agents* and the target agent Ag_b.

Following the ideology that Ag_a could, to some certain extent, rely on its own history interaction with Ag_b and partially use the second approach which is consulting some other agents, Ag_a gives a 100% trustworthy rate to its history and use it as a portion in its trustworthy community. This merging method takes into account the proportional relevance of each trust value, rather than treating them separately. Basically, the contribution percentage is set regarding to how informative the history is in terms of the number of direct interactions from the history. Therefore, contribution is higher if the history represents a lower entropy. Respectively, the higher entropy makes less rely on the history and thus the new evaluation is more considered. The mentioned entropy is also affected by the coherency of the quality of the service provided by the agent in question. If the belief about any agent is updated by the rates corresponding to the quality of provided service with a very low deviation, then the history is considered more reliable. However, the new evaluation is merged by the previous data and we tend to analyze the quality of the service of the target agent regarding to what is expected and what is actually provided. Likewise the Ψ function indicates the similar relative coefficients regarding to the corresponding *referee agents*.

Equation (1) takes into account the three most important factors: (1) the trustworthiness of *trustworthy/referee agents* according to the point of view of Ag_a ($Tr_{Ag_a}^{Ag_i}$ and $Tr_{Ag_a}^{Rf_j}$); (2) the Ag_b's trustworthiness according to the point of view of *trustworthy/referee agents* ($Tr_{Ag_i}^{Ag_b}$ and $Tr_{Rf_j}^{Ag_b}$); (3) the number of interactions between these *trustworthy/referee agents* and Ag_b ($N_{Ag_b}^{Ag_i}$ and $N_{Ag_b}^{Rf_j}$).

5.4 Proof of Concepts

In this section, we assess the CRM model efficiency and implement a proof of concept prototype. In this prototype, agents are implemented as $Jadex^{©TM}$ agents (Section 3.3). The agent reasoning capabilities are implemented as Java modules using logic programming techniques. All java classes, objects and methods are described in Section 3.3 and Appendix 1. Each agent has a knowledge base about the reputation of other agents, as hashtables object in java. Such a knowledge base has the following structure: $Agent - name$, $Agent - reputation$, $Total - interaction - number$ and $Recent - interaction - number$. The visited agents during the evaluation process are updated in the $Jadex^{©TM} beliefsets$. We have a manager agent who decides which agent should be in which agent's radius of activity and the agent with radius of activity sets its knowledge base for those it knows or are in its circle in the beginning of the simulation. One simulator agent decides the number of runs and asks all agents to provide cumulative utility at the end of each run. Also, there is a selector agent who selects randomly some agents from the directory facilitator [37] where all agents are described and re registered.

	S.P. Agent Type	Density in the S.P. Community	Provided Utility at Each RUN		Radius of Activity
			Range	Standard Deviation	
Service Provider Agents (S.P.)	Good	15.0%	[+5, +10]	1.0	25
	Ordinary	30.0%	[-5, +5]	2.0	28
	Bad	15.0%	[-10, -5]	2.0	25
	Fickle	40.0%	[-10, +10]	-	30
	S.C. Agent Type	Density in the S.C. Community	Number of Joining Agents at Each RUN		Radius of Activity
Service Consumer Agents (S.C.)	CRM	25.0%	6		35
	FIRE	25.0%	6		35
	REFERRAL	25.0%	6		35
	SPORAS	25.0%	6		35

Table 5.1 Protocol minimization over the obtained measurement

The testbed environment (represented in table 5.1) is populated with two type of agents; service provider agents who are mend to provide services (toward simplicity, we assume only one type of service is provided and therefore consumed) and service consumer agents (equipped with the aforementioned trust model) who are seeking the service providers to interact with and consume the provided service and therefore gain the corresponding utility. This utility depends on type of the service provider. Generally, service providers are different, and thus they provide different quality of service and the consumer agents who use these services obtain diverse utility.

Each agent (either service provider or consumer) is located randomly over the environment and has been assigned a radius of activity in which it is centralized and be known by all other agents who are in the area of activity. This simply means agents who are close enough together, have private belief about each other. However, this does not exclude the fact that agents extend their activity area and gradually get acquainted to other agents who are not in their activity area.

The simulation consists of a number of consequent RUNs in which agents are activated and build their private knowledge and keep interacting with one another, gain utility and enhance their overall knowledge about the environment. The more agent knows the environment, the better it can choose service providers and thus the more utility it gains. Agents are free to ask others of their belief about the service provider to be selected. Finally, each agent requests for service from the provider that the agent found the most trustworthy and reliable. This does not mean that the agent can expect a certain utility from the selected service provider and the service provider is more or less flexible in the quality of the service being provided. Table 5.1 represents four types of the service providers: good, ordinary, bad and fickle. The first three provide the service regarding to the assigned mean value of quality with a small range of deviation. However, fickle providers are more flexible as their range of quality covers the whole possible outcomes. To put the system in a tighter situation, we gave a high number of fickle agents.

Since the major difference between frameworks is the trust model they employ for credibility assessment, the utility gained by each model is considered as its efficiency in selecting reliable service providers. Doing so, we compare CRM with three other models (FIRE, a successful trust model with high performance [69], SPORAS, which is a centralized approach [72] and Referral, which follows the concept of references [73]) in an honest environment to be able to represent the comparison illustrated in related works. Moreover, we carry on comparing

CRM with FIRE model in more details in a biased environment in which CRM agents expose a higher efficiency where the change of behavior is an issue.

5.4.1 Honest Environment

Figure 5.4 depicts the overall comparison of different models; The testbed consists of a number of RUNs and consumer agents get service from the service provider agents after evaluating trust and decided to interact; this number of interactions is represented as the horizontal axes, and the mean value ranking for the utility gained of each group are represented in the vertical axes. As the RUNs are elapsing, each service consumer is using a particular model to find the most trustworthy service provider and thus gain the most. The utility gained means of agents using the same trust models are compared with each other's using two sample t-test with 95% of confidence level represented in the ranking form to show the overall outperforming of CRM and FIRE comparing to the other two.

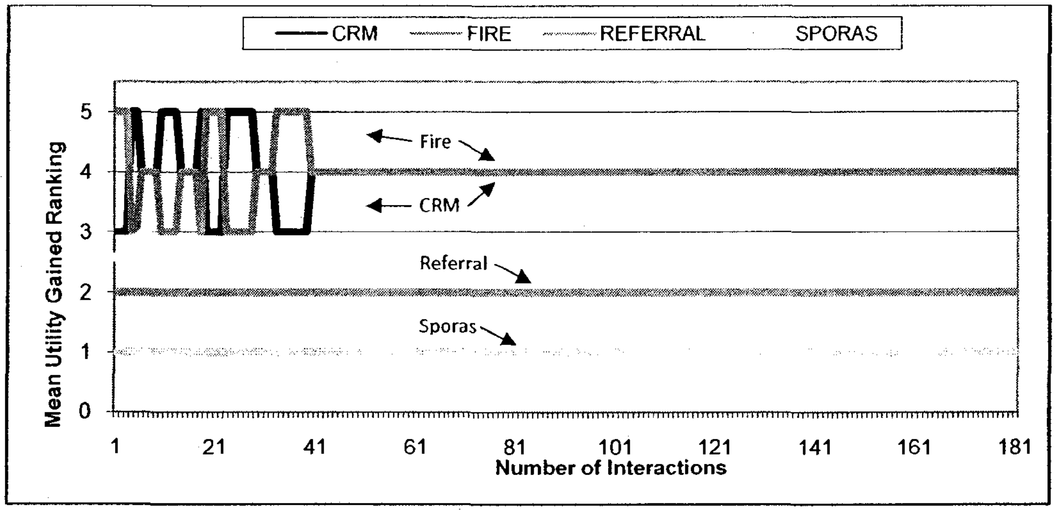


Figure 5.4 Comparison of CRM with FIRE, Referral and Sporas model, in terms of mean utility gained value at each RUN

Groups reflect the performance of four different trust models we considered for comparison. SPORAS system is known as independently developed model which is generally used as benchmarks. Since SPORAS is a centralized model, it suffers from inconsistency of the trust values associated to agents, while they register upon entrance. Thus this system would not perform well in situations when the good service providers are new to the system and remain unknown for longer time comparing to others. Relatively we still have the problem of fake advertising to the central agent to get more benefit. Therefore, SPORAS performs weak in selecting the best service providers. Referral model agents directly consider how to place trust in others and emphasize the key properties that affect the trust assessment, however they do not take into account the suggestions of other agents, which lead them to assess the credibility of an unknown or partially known service provider. This may affect the selection of good providers from the beginning of the simulation. FIRE agents [69], regulates the problem of collecting the required information by the evaluator to assess the trust of his partner. In addition they apply certified reputation introduced by the target agent. As results of t-test illustrated in Figure 5.4, the commutative utility gained over the 500 elapsed RUNs by FIRE and CRM agents are culminated to be the highest as both methods select good service providers and therefore gain the highest possible utility. In this environment the agents are considered to be honest and they reveal their belief with 100% accuracy. In the next section, we carry on by the biased environment in which agents would not necessarily reveal with 100% accuracy and this cause the evaluator agent to be confused in the trust assessment and we discuss how CRM agents cope with such a problem.

5.4.2 Biased Environment

Being more realistic, we exposed the same agents in a very biased environment in which the agents, serving some certain goals, may reveal much less accurate information. Each agent accumulates the utility gained along interactions taken place employing its corresponding trust model. We continue the comparison with FIRE model. Experimental variables are outlined in table 5.2 and illustrated in figure 5.5. In order to perform an accurate comparison between aforementioned trust models, each model is used by 50 consumer agents who seek service from total 20 service providers providing diverse range of utility. In each RUN, 12 agents are joined to seek for the best service provider and objectively gain the highest possible utility.

Measurements and Characteristics	CRM	FIRE
No. of active agents in simulation	50	50
No. of RUNs in each simulation	500	500
Measured cumulative utility gained in five simulations	11,947	8,429
	9,445	8,063
	7,408	4,652
	11,440	5,538
	9,432	9,092
Average cumulative utility gained	*9,534*	*6,554*
Standard deviation of cumulative utility gained	1,624	1,837
Half value of confidence interval	1,710	1,934
Full confidence interval	**(7,824 - 11,244)**	**(4,620 - 8,488)**

Table 5.2 Protocol minimization over the obtained measurement

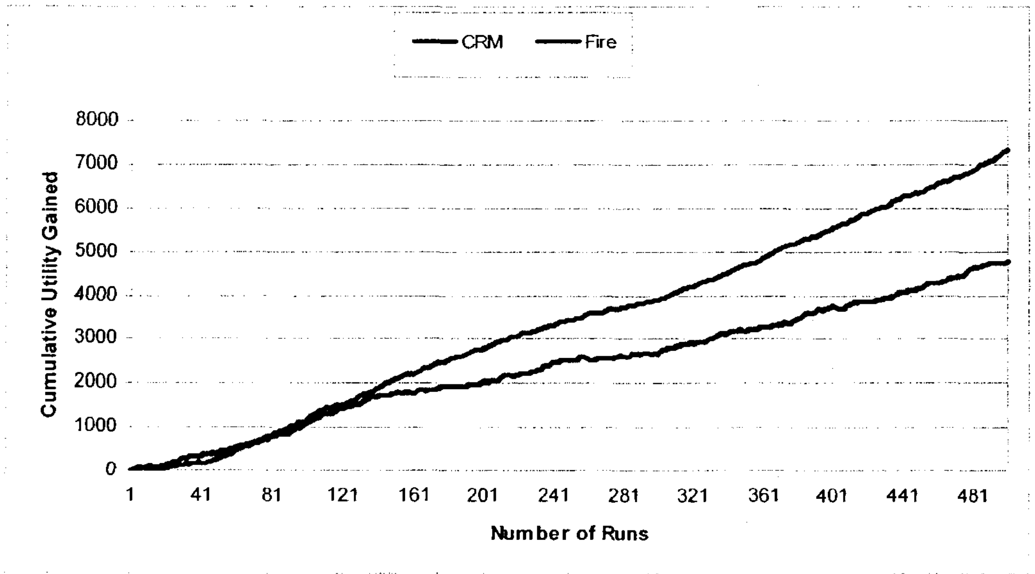


Figure 5.5 Comparison of CRM with FIRE model, in terms of commutative utility gained value over the RUNs.

In this case, FIRE agents collect the information gained by other agents and the target agent to assess the credibility. However, they do not recognize the spurious ratings generated by some malicious agents; and in a biased environment these agents quickly fail and drop their accuracy in credibility estimation, which leads them to regular selection of fickle service providers. We left the discussion of the CRM and FIRE agents for the next section in which we discuss the advantages of CRM in more details by presenting an experimental scenario. In some cases agents do not propose a good *referee agent* and as a rational agent, it picks up the referee who is more beneficial for him rather than the system, thus in this case the final trust rate would be affected with non-reliable information about the target agent. Eventually the agents imagination about the target agent will not be true, therefore the evaluating agent has to evaluate the *referee agents*, although it will cost an extra computational overhead for the method.

5.5 Experimental Results

FIRE is a successful trust-certified reputation model which addresses the problem of lack of direct history. Agents evaluate the trust of other agents as a decentralized service. However FIRE agents do not recognize the agents who got the good ratings and performed bad either in terms of the inaccurate ratings provided for some others or unacceptable utility provided. CRM agents are equipped wiih protocol which enables them to recognize change of behavior of others and respectively adjust their beliefs regarding to the functioning of some particular known agents. This basically states the collusion problem, by which agents intentionally reveal non-accurate information, aiming to gain more benefit at the end. This change of behavior should be recognized and the benefit of other agents should get adjusted for the new manner of the changed agent. This process also quickly recognizes the fickle agents who may provide any quality of service. Figure 5.6 illustrates a scenario in which an agent in collision with some other agents tries to ignore a typical service provider.

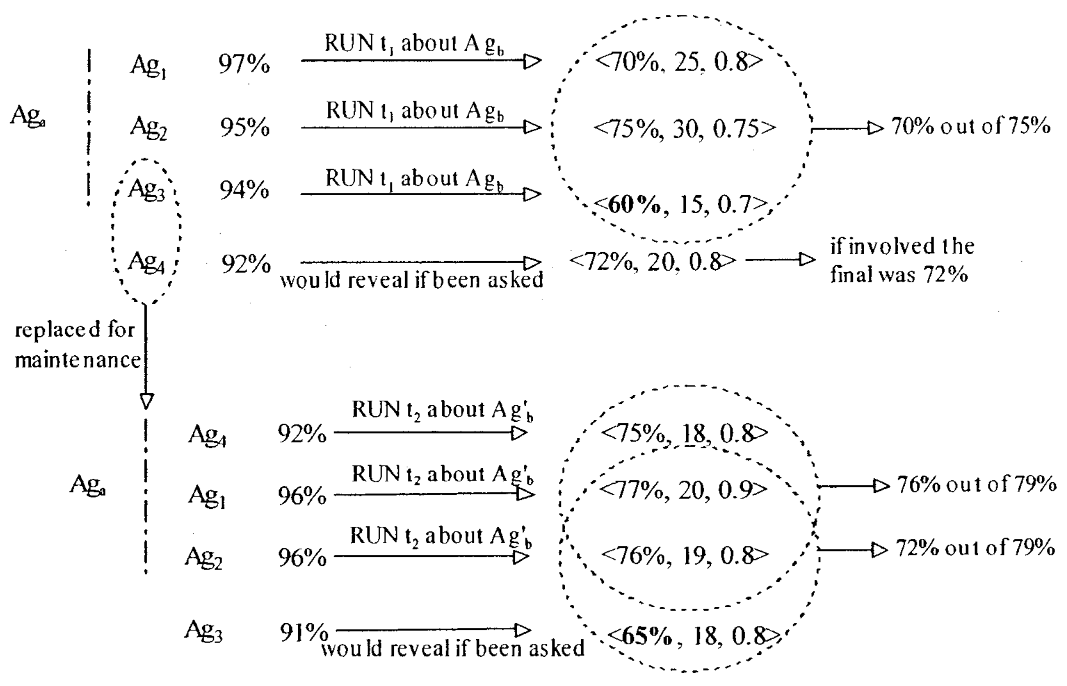

Ag_a
Ag_1 97% RUN t_1 about Ag_b <70%, 25, 0.8>
Ag_2 95% RUN t_1 about Ag_b <75%, 30, 0.75> 70% out of 75%
Ag_3 94% RUN t_1 about Ag_b <60%, 15, 0.7>
Ag_4 92% would reveal if been asked <72%, 20, 0.8> if involved the final was 72%
replaced for maintenance
Ag_a
Ag_4 92% RUN t_2 about Ag'_b <75%, 18, 0.8>
Ag_1 96% RUN t_2 about Ag'_b <77%, 20, 0.9> 76% out of 79%
Ag_2 96% RUN t_2 about Ag'_b <76%, 19, 0.8> 72% out of 79%
Ag_3 91% would reveal if been asked <65%, 18, 0.8>

Figure 5.6 Substituting untrustworthy agent in maintenance step

Suppose Ag_a after a period of time decides to balance the credibility of the other consulting agents who had very recently revealed some information regarding to Ag_b's credibility. From Ag_a's point of view, a consulting agent has revealed accurate information when it is close enough to the actual performance of Ag_b, and oppositely a consulting agent is known to be not accurate when the provided belief is apart from what has been seen by Ag_a. Suppose at RUN t_1, Ag_a asks the already defined set of trustworthy (Ag_1 with 97% of credibility, Ag_2 with 95% of credibility and Ag_3 with 94% of credibility,) and *referee agents* their belief about Ag_b. Ag_1 discloses 70% to be the Ag_b's trustworthiness based on 25 interactions which are valid 0.8 of time recency. Respectively, Ag_2 discloses the required information as 75%, 30 interactions by 0.75 of time recency and Ag_3 provides 60%, 15 interactions and 0.7 time recency. After evaluation process of Ag_a, Ag_b is known to be 70% trustworthy, but in reality after a period of interaction, Ag_b shows 75% accuracy in the service provided.

Let us discuss the same scenario when Ag_4 (known to be 92% trustworthy) was involved as a *trustworthy agent* (instead of Ag_3), and upon Ag_a's request, Ag_4 would provide 72% of credibility based on 20 interactions by 0.8 of time relevance. Considering the number of interactions and the accuracy of provided information, Ag_4 seems to be more acquainted with Ag_b rather than Ag_3, therefore the choice of $\{Ag_1, Ag_2, Ag_4\}$ could have been a better choice of trustworthy community to ask about Ag_b in which the evaluation process would end up with the final value of 72% which is closer than previous estimation.

The objective of the maintenance that CRM model performs is to overcome this type of inefficiencies. Based on formula 14 of [21], the trustworthy of Ag_3 would drop to 91% which automatically put Ag_4 at a higher rate of trustworthy. Now performing a new evaluation process done by Ag_a about $Ag_{b'}$ who in reality performed 75% of accuracy in the provided services, consider two cases of with and without maintenance. In the first case, Ag_4 would be replaced by Ag_3, thus Ag_a would request the new trustworthy community ($\{Ag_1, Ag_2, Ag_4\}$) their belief about $Ag_{b'}$; in this case Ag_1, Ag_2 and Ag_4 respectively would respond 77%, 76% and 75% and finalize the evaluation of the $Ag_{b'}$ to be 76% which seems to be fairly close to the real credibility of $Ag_{b'}$. In the second case, Ag_a's trustworthy community is still $\{Ag_1, Ag_2, Ag_3\}$, and upon request, Ag_1, Ag_2 and Ag_3 respectively would respond 77%, 76% and 65% and finalize the evaluation of the $Ag_{b'}$ to be 72%. This value is affected because of the participation of Ag_3 who has recently started malfunctioning.

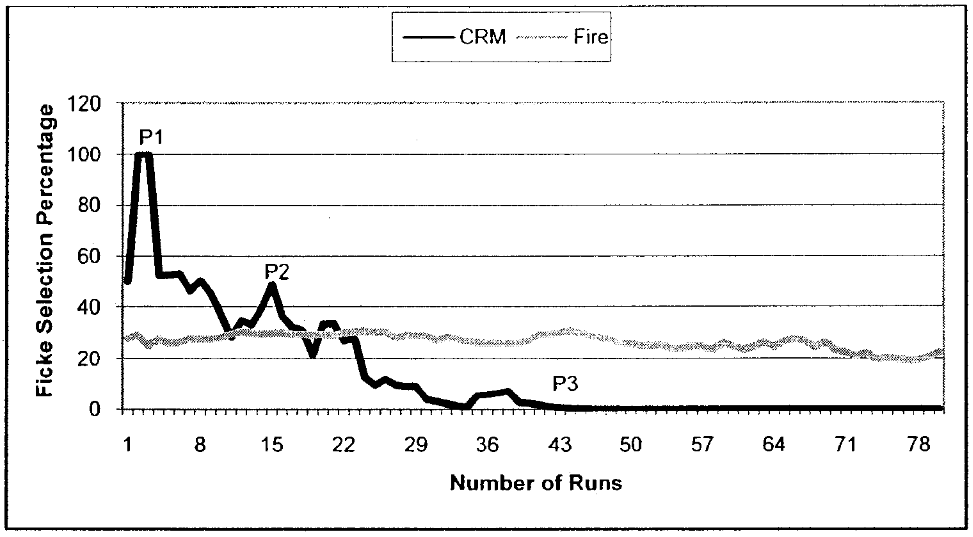


Figure 5.7 Comparison of CRM and FIRE Model in terms of selecting fickle service providers along the elapsing RUNs

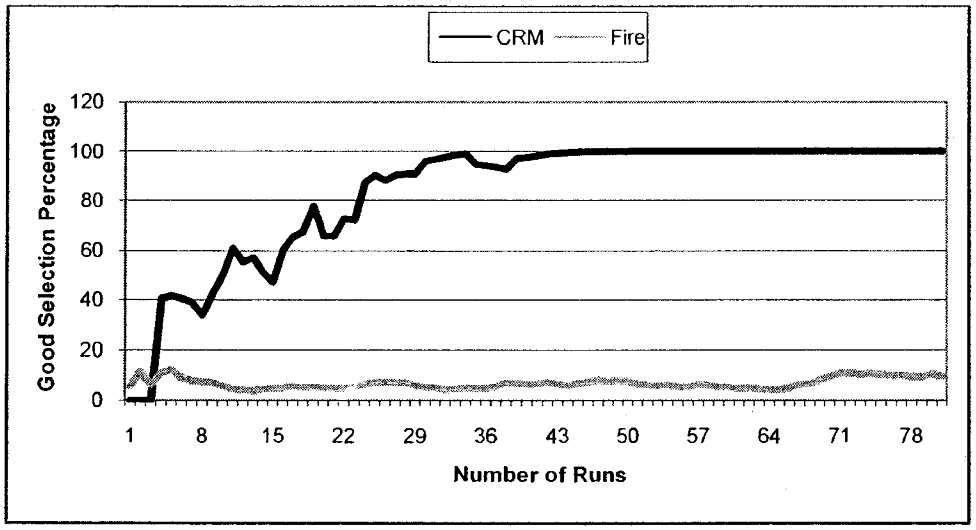


Figure 5.8 Comparison of CRM and FIRE Model in terms of selecting good service providers along the elapsing RUNs

Figure 5.7 shows a graph plotting fickle selection percentage versus number of RUNs. The graph highlights the difference of having and missing the maintenance regarding to the behavior of CRM and FIRE agents. In the first 80 RUNs, we observed that CRM agents are reducing the selection of fickle agents in the RUNs as the time goes on. This is because the CRM agents would perform maintenance on the behavior of the fickle agents who provide a bad utility after the interaction and deduce their belief about them which leads to less selection afterwards. Relatively FIRE agents would almost remain same as they do not recognize the fluctuated behavior of the fickle agents. The picks of the CRM graph (P_1 and P_2) are simply because of selection of few number of CRM agents at each RUN and therefore the maintenance they perform would generally has low affect on the consequent RUN until they are selected again. Therefore, the curve would come down in a fluctuated manner until all the fickle agents lose their credibility and never get selected which happens in P_3. Respectively Figure 5.8 illustrates the same type of the graph with the good agent selection percentage versus the number of RUNs. This graph is the complementary of the graph represented in Figure 5.7 as the less fickle providers are selected, the more good providers are recognized and therefore, CRM agents would enhance their credibility and after distribution of the obtained ranking good providers are always selected.

5.6 Related Work

Perhaps the best-known approaches to trust in multi-agent systems are FIRE [69], SPORAS [72] and Referral [73]. In this section, in addition we get more into details by analyzing some recently emerged systems like ReGret [74], Formal [75], HIT [76], Adaptive [77] and Statistics [78]. So far the proposed approaches are distinguishable by the following classifications: 1) Policy-based trust; 2) Reputation-based trust; 3) General model of trust; and 4) Trust in information resources. Generally speaking, all the approaches are following a direction to overcome the following problems: The model should be provided by adequate information related to the environment and the contributing agents; they tend to avoid consulting with a central control unit who is always subject to single point of failure or huge bottleneck (for example in online auction development). Agents are aimed to make estimation independently; there are always malicious agents who try to distract the overall process; they can either try to slander other agents by lying about its trust level or supporting an agent on purpose, try to exaggerate about its credibility.

The idea of witness reputation has been used by Sabater who proposed a decentralized trust model [74] called ReGret. He used the reports from the witnesses in addition to the technique based on direct interaction experience. One of the substantial aspects of this work is unlike the previous approaches, the rating are dealt according to their recently relevance. Thus, old ratings are given less importance compared to new ones. Sabater's work is sensitive to noise and thus vulnerable as it does not represent witness locations. Also, it does not notice distractions made by some malicious agents. In our model, the issue is managed by considering the witnesses trust and our merging method takes into account the proportional relevance of each reputation value, rather than treating them equally.

Singh in the other work with Wang developed as algebra [75] for aggregating trust over graphs understood as webs of trust. They argue that current approaches for combining trust reports tend to involve ad hoc formulas. In their work, dynamism is accommodated by discounting over time and composition by discounting over the space source. They have developed a principled evidential trust model that would underlie any such agent system where trust reports are gathered from multiple sources.

Regarding to ad hoc formulation, Velleso et al. presented a similar work applied to ad hoc network [76]. The aspect of their work is that they have refereed to human concept of trust. Similar to our work they use the recommendations by *trustworthy agents* in addition to their own direct experience. They tried to balance the recommendations regarding to recently relevance and relationship maturity, but the agents do not have reasoning capabilities, moreover they do not have policies taken for dealing with the malicious agents.

Song, Phoha and Xu, proposed an Adaptive recommendation trust model [77] for multi-agent systems. They design a neural network for evaluating multiple recommendations of various trust standards with and without deceptions. They used an ordered depth first search (DFS) for delaying the first initial set of qualified recommendations (preparing a proper data set for proposed neural network input). In the second stage they design a neural network which is based on back propagation. The output of this stage will be the actual set of qualified recommendations. The most important advantage of this model is adaptively and flexibility that captures the dynamic nature of online trust. On the other hand using neural network in dynamic environment needs much more time for training faze of neural net, thus when our input data set has changed our designed neural net must be adapted and it needs a large amount of time considering time period for each iteration in Multi-Agent Systems. As each trust model needs to update its recommendations and we have to consider the time relevance factor in recommender qualification faze of our system, designed neural network must be run frequently and it causes time complexity overhead. On the other hand there is no method in their proposed approach to solve the report refusal problem and there is no chance for the target agent to introduce his *referee agents* to us and these flaws cause a late convergence problem for neural network or may be in accurate trust estimation.

In the work proposed by Shi et al. [78], a trust model has been introduced to assist decision-making in order to predict the likely future behavior by analyzing the past behavior. The authors have mostly worked on the environment facilitation, for example the space of possible outcomes has been studied. They believe it is crucial to identify the space of possible outcomes which determines the nature of the associated trust model. The notion of discrete categories is similar to our model in terms of giving more flexibility to the ratings as feedback in order to get more accurate direct interaction estimation. But they have not taken into account the measurements which would unbalance the trust estimation and their decision-makings are solely based on the previous interactions but in our model after a certain amount of time a maintenance is performed to dynamically update the policies adopted.

Chapter 6. Conclusions and Future Work

6.1 Contributions and Concluding Remarks

In this dissertation, a 3-level framework for *B2B* applications was presented. The three levels namely strategic, application and resource are populated with argumentative agents. We have shown how our framework can be used to set up collaborations among autonomous businesses (via strategic level), and execute and manage these collaborations (via application level). Inevitably, given the autonomous nature of businesses, conflicts are bound to arise. We have shown how our framework can detect and resolve conflicts. To this end an argumentation-based model was developed and implemented. This model was the basis of a negotiation and persuasion protocol that includes inquiry stages for resolving conflicts between agents acting on behalf of applications of type Web services. This protocol has the originality of considering partial arguments allowing agents acting on behalf of businesses to reason about partial information.

Another contribution of this thesis is the proposition of a new probabilistic and statistic-based model to secure open multi-agent systems such as the one proposed and developed in the first part of this dissertation, which is about *B2B* applications. In this system, agents communicate with each other using dialogue games. A framework based upon trustworthy and referee agents has been presented. Furthermore, this framework considers many machanisms allowing agents to make use of the information communicated to them by other agents to determine the trust of further target agents. Our model has the advantage of being computationally efficient and of taking into account three important factors: (1) the trust (from the viewpoint of the evaluator agent) of the trustworthy agents; (2) the trust value assigned to target agent according to the point of view of trustworthy agents; and (3) the number of interactions between trustworthy agents and the target agent. Moreover, agents perform an off-line maintenance in order to evaluate the consulting agents' trust level by comparing the provided information regarding to the target agent's trust level and the actual behavior of the target agent since it has started interaction. The resulting model allows us to produce a comprehensive assessment of the agents' credibility in a software system. The simulations we carried out have shown the efficiency of the proposed model compared to the existing models in the literature.

6.2 Future Work

For future work, we plan to investigate the following points:

1. Propose a general framework for agent negotiation by considering the formalization of concessions and there effects on the outcome of negotiation protocols. Computational argumentation theory provides a promising base for understanding and modeling concessions by analyzing the strength of exchanged arguments and by building new arguments when new information become available.

2. Analyze and enhance the computational complexity of the proposed framework. The complexity of deciding if an argument is a valid one and if an agent can build arguments for given conclusions is high in propositional languages. However, considering less general languages that are enough to express agent beliefs can resolve this problem.

3. Scale up and demonstrate our argumentation-based model on larger examples. Additionally, we plan to enrich our model with contextual ontologies when modeling knowledge bases of individual agents.

4. Consider the effect of using argumentation reasoning when assessing the trust of other agents. Indeed, when interacting, agents are not always using quantitative methods to evaluate the provided service. However, they can argue about the quality of this service. In addition, when

asking testimonies for their opinions about other agents, argumentation can play a fundamental role since quantitative evaluation can differ from an agent to another. By showing the arguments that are used in the evaluation, agents can have a sophisticated reasoning to accept or refuse the testimony. In fact, agents are not supposed to be always honest when sending their opinions to others. By using an argumentation reasoning their strategies can change, and they will not simply asking others, but try to argue with them before taking a decision. Furthermore, merging trust and argumentation in a combined and unified framework will solve many problems in trust evaluation that are generally based on heuristics.

References

[1] G. Weiss; Multiagent systems a modern approach to distributed artificial intelligence; MIT Press, 1999.

[2] M. Wooldridge; An introduction to Multiagent Systems; J.Wiley, 2002.

[3] A. Rogers, E. David, J. Schiff and N.R. Jennings; The Effects of Proxy Bidding and Minimum Bid Increments within eBay Auctions; ACM Transactions on The Web, Vol. 1, No. 2, Article 9, August 2007

[4] N. Schurr, J. Marecki, M. Tambe and P. Scerri; The Future of Disaster Response: Humans Working with Multiagent Teams using DEFACTO; AAAI Spring Symposium on Homeland Security, 2005.

[5] R. Sun and I. Naveh; Simulating Organizational Decision-Making Using a Cognitively Realistic Agent Model; Journal of Artificial Societies and Social Simulation vol. 7, no. 3.

[6] M. Dastani, J. Hulstijn, and L.V. der Torre; Negotiation protocols and dialogue games; Artificial Intelligence Conference, pp. 13-20, (2000).

[7] N.C. Karunatillake, N.R. Jennings, I. Rahwan and T.J. Norman; Argument-based negotiation in a social context; AAMAS Conference, pp. 1331-1332, (2005).

[8] C. Li, J.A.Giampapa and K.P.Sycara; Bilateral negotiation decisions with uncertain dynamic outside options; IEEE Transactions on Systems, Man, and Cybernetics, Part C 36(1): 31-44, (2006).

[9] I. Rahwan, L. Sonenberg, N. R. Jennings and P. McBurney; STRATUM: A Methodology for Designing Heuristic Agent Negotiation Strategies; Applied Artificial Intelligence, Vol 21, No 6, pp. 489-527. (2007).

[10] H. Prakken; Relating protocols for dynamic dispute with logics for defeasible argumentation Syntheses; pp. 187-219, (2001).

[11] P. McBurney, S. Parsons, and M. Wooldridge; Desiderata for Agent Argumentation Protocols; AAMAS Conference, pp. 402 – 409, (2002).

[12] F. H. van Eemeren and R. Grootendorst; Argumentation, Communication and Fallacies: A Pragma-Dialectical Perspective; (1992).

[13] L. Amgoud, S. Belabbes, and H. Prade; A Formal General Setting for Dialogue Protocols; AIMSA, pp. 13-23, (2006).

[14] J. Bentahar, B. Moulin, and B. Chaib-draa; Commitment and argument network: a new formalism for agent communication; pp. 146-165, (2003).

[15] J. Bentahar, B. Moulin, and B. Chaib-draa; Specifying and Implementing a Persuasion Dialogue Game using Commitment and Argument Network. Argumentation in Multi-Agent Systems; vol. 3366 of LNAI, pp. 130-148, (2005).

[16] P. McBurney and S. Parsons; Dialogue Games in Multi-Agent Systems Informal logic; Special Issue on Applications of Argumentation in Computer Science, vol. 22(3): 257-274, (2002).

[17] J. Bentahar, J. Labban; An Argumentation-Driven Model for Autonomous and Secure Negotiation; GDN Conference, pp. 5-18, (2007).

[18] I. Rahwan, S. Ramchurn, N. R. Jennings, P. McBurney, S. Parsons and L. Sonenberg; Argumentation - based negotiation; The Knowledge Engineering Review, Vol.18:4,pp.343-375, Cambridge University Press (2004).

[19] W. Shen, D. H. Norrie and Jean-Paul Barthès; Multi-agent Systems for Concurrent Intelligent Design and Manufacturing; CRC press(2001)

[20] Virtual Roundtable, Internet Computing on-line Journal, July-August issue, 1997.

[21] B. Khosravifar, J. Bentahar, M. Gomrokchi and R. Alam; An approach to Comprehensive Trust Management in Multi-Agent Systems with Credibility; IEEE 3rd International Conference on Research Challenges in Information Sciences, RCIS, Marrakech, Morocco 2008.

[22] N. Kasabov; Introduction: Hybrid intelligent adaptive systems; International Journal of Intelligent Systems, Vol.6, (1998) 453-454.

[23] N. R. Jennings and M. Wooldridge; Agent-Oriented Software Engineering; Handbook of Agent Technology (ed. J. Bradshaw) AAAI/MIT Press, 2000.

[24] S. A. DeLoach, M. F. Wood and C. H. Sparkman; Multiagent Systems Engineering; The Intl. Jour. of SE and KE, Vol. 11, No. 3, June 2001.

[25] P. Bayer and M. Svantesson; Comparison of Agent-Oriented Methodologies Analysis and Design, MAS-CommonKADS versus Gaia; Blekinge Institute of Technology, Student Workshop on Agent Programming, 2001.

[26] M. A. Ardis, J. A. Chaves, L. J. Jagadeesan, P. Mataga, C. Puchol, M. G. Staskauskas and J. Von Olnhausen; A Framework for Evaluating Specification Methods for Reactive Systems, Experience Report; IEEE Trans. Software Engineering, Vol. 22, No. 6, pp 378-389, June 1996.

[27] M. Bunge; Treatise on Basic Philosophy: Vol. 3, Ontology I: The Furniture of the World; Reidel, Boston, June 1977.

[28] M. T. Cox and S. DeLoach; Multiagent systems and mixed initiative planning course; Wright State University http://www.cs.wright.edu /people/faculty/mcox/ Teaching/Cs790/, April 2000.

[29] L. Cernuzzi and G. Rossi; On the Evaluation of Agent Oriented Methodologies; in Proc. of the OOPSLA 2002 Workshop on Agent-Oriented Methodologies, November 2002.

[30] M. Wooldridge, N. R. Jennings, and D. Kinny; The Gaia Methodology for Agent-Oriented Analysis and Design; Journal of Autonomous Agents and Multi Agent Systems, Vol. 3, No. 3, pp. 285-312, March 2000.

[31] C. A. Iglesias, M. Garrijo, J. Gonzalez , and J.R.Velasco; Analysis and Design of multiagent systems using MAS-CommonKADS; Proceedings of the Fourth International Workshop on

Agent Theories, Architectures and Languages (ATAL), LNCS 1365, Springer-Verlag, pp. 313-328, 1998.

[32] E. Yu and L.M. Cysneiros; Agent-Oriented Methodologies–Towards A Challenge Exemplar; 4th Intl. Workshop on Agent-Oriented Information Systems (AOIS'02), May 2002.

[33] C. A. Iglesias, M. Garijo, and J. C. Gonzalez; A survey of agent-oriented methodologies; Intelligent Agent V, Proc. of ATAL-98, LNAI 1555, pp. 317-330, Springer, July 1999.

[34] P. Bresciani, P. Giorgini, F. Hiunchiglia, J. Mylopoulos, and A. Perini; Tropos: An agent-oriented software development methodology; AAMAS Journal, 2004; 8(3): 203-236.

[35] A.S. Rao and M.P. Georgeff; Modeling Rational Agents within A BDI- Architecture; Second Int'l Conf. on Principles of Knowledge Representation and Reasoning (KR'91), Morgan Kaufmann: San Mateo, 1991, pp. 473-484.

[36] C.E. Lin, M. Krishna, T. Fredrick and M. Kris; A Methodology to Evaluate Agent Oriented Software Engineering Techniques; proceeding of the 40th Hawaii International Conference on system science – 2007.

[37] Jadex User Guide http://vsis-www.informatik.uni-hamburg.de/projects/jadex/

[38] L. Amgoud, Y. Dimopoulos, and P. Moraitis ; A Unified and Genaral Framwork for Argumentation based Negotiation; In The International Conference on Autonomous Agents and Multiagent Systems. ACM Press, 2007.

[39] K. Atkinson, T. Bench-Capon, and P. McBurney; A Dialogue Game Protocol for Multi-Agent Argument over Proposals for Action; Journal of Autonomous Agents and Multi-Agent Systems, 11(2), 2005.

[40] J. Bentahar, Z. Maamar, D. Benslimane, and P. Thiran; An Argumentation Framework for Communities of Web Services; IEEE Intelligent Systems, 22(6), 2007.

[41] J. Bentahar, F. Toni, J.-J. Meyer, and J. Labban; A Security Framework for Agent-based Systems; Journal of Web Information Systems, 3(4):341–362, 2007.

[42] E. Black and A. Hunter; A Generative Inquiry Dialogue System; In The International Conference on Autonomous Agents and Multiagent Systems, ACM Press, 2007.

[43] A. Bondarenko, P. Dung, R. Kowalski, and F. Toni; An Abstract, Argumentation-Theoretic Approach to Default Reasoning; Artificial Intelligence, 93(1–2):63–101, 1997.

[44] P. Dung; The Acceptability of Arguments and its Fundamental Role in Non-Monotonic Reasoning and Logic Programming and n-Person Game; Artificial Intelligence, 77:321–357, 1995.

[45] P. Dung, P. Mancarella, and F. Toni; Computing ideal sceptical argumentation; Artificial Intelligence, Special Issue on Argumentation in Artificial Intelligence, 171(10-15):642–674, 2007.

[46] H. Fujita; Special Issue on Techniques to Produce Intelligent Secure Software; Knowledge Based Syst., 20(7):614–616, 2007.

[47] A. Garcia and G. Simari; Defeasible Logic Programming: an Argumentative Approach; Theory and Practice of Logic Programming, 4(1):95–138, 2004.

[48] V. Gruhn and H. Fujita; Special Issue on Intelligent Software Design; Knowledge Based System, 19(2):105–106, 2006.

[49] B. Ktari, H. Fujita, M. Mejri, and D. Godbout; Toward a New Software Development Environment; Knowledge Based System, 20(7):683–693, 2007.

[50] R. Lau; Towards aWeb Services and Intelligent Agents-based Negotiation System for B2B eCommerce; Electronic Commerce Research and Appl., 6(3), 2007.

[51] N. Maudet and B. Chaib-draa; Commitment-based and Dialogue Game-based Protocols, new trends in agent communication languages; Knowledge Engineering Review, 17(2):157–179, 2002.

[52] P. McBurney and S. Parsons; Games that Agents Play: A Formal Framework for Dialogues between Autonomous Agents; Journal of Logic, Language, and Information, 11(3):315–334, 2002.

[53] N. Milanovic and M. Malek; Current Solutions for Web Service Composition; IEEE Internet Computing, 8(6), November/December 2004.

[54] C. Nicolle, K. Y'etongnon, and J. Simon; XML Integration and Toolkit for B2B Applications; Journal of Database Management, 14(4), 2003.

[55] S. Parsons, M. Wooldridge, and L. Amgoud; On the Outcomes of Formal Inter-Agent Dialogues; In Proceedings of The International Conference on Autonomous Agents and Multiagent Systems. ACM Press, 2003.

[56] P. Pasquier, I. Rahwan, F. Dignum, and L. Sonenberg; Argumentation and Persuasion in the Cognitive Coherence Theory; In The 1st International Conference on Computational Models of Argument. IOS Press, 2006.

[57] H. Prakken; Formal Systems for Persuasion Dialogue; The Knowledge Engineering Review, 24(2), 2005.

[58] F. Sadri, F. Toni, and P. Torroni; Dialogues for Negotiation: Agent Varieties and Dialogue Sequences; In The International Workshop on Agents, Theories, Architectures and Languages, 2001.

[59] Y. Udupi and M. Singh; Contract Enactment in Virtual Organizations: A Commitment-Based Approach; In The 21st National Conference on Artificial Intelligence, 2006.

[60] D.N. Walton and E.C.W. Krabbe; Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning; Suny Series in Logic and Language, 1995.

[61] A. Williams, A. Padmanabhan and B. Blake; Local Consensus Ontologies for B2B-Oriented Service Composition; In The International Joint Conference on Autonomous Agents and Multiagent Systems, 2003.

[62] http://www.doc.ic.ac.uk/~dg00/casapi.html

[63] http://sourceforge.net/projects/argkit/

[64] http://jlogic.sourceforge.net/

[65] http://sourceforge.net/projects/tuprolog/

[66] J. Bentahar and J-J. Ch. Meyer; A new quantitative trust model for negotiating agents using argumentation; In the International Journal of Computer Science and Applications,4(2):1-21, 2007.

[67] P. Yolum and M.P. Singh; Engineering self-organizing referral networks for trustworthy service selection; IEEE Transaction on systems, man, and cybernetics, 35(3):396-407, 2005.

[68] E. Shakshuki, L. Zhonghai, and G. Jing; An agent-based approach to security service. International Journal of Network and Computer Applications; Elsevier, 28(3): 183-208, 2005

[69] T. Dong-Huynh, N.R. Jennings and N.R. Shadbolt; Fire: An integrated trust and reputation model for open multi-agent systems; Journal of Autonomous Agents and Multi-Agent Systems 13(2) pp. 119-154, 2006.

[70] T. Dong-Huynh, N.R. Jennings and N.R. Shadbolt; Certified reputation: How an agent can trust a stranger; In Proceedings of The Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 1217-1224, Hakodate, Japan, 2006.

[71] T. D. Huynh, N. R. Jennings, and N. R. Shadbolt; An integrated trust and reputation model for open multi-agent systems; Journal of Autonomous Agents and Multi-Agent Systems AAMAS, 2006, 119-154.

[72] G. Zacharia, and P. Maes; Trust management through reputation mechanisms; Applied artifitial intelligence, 14(9):881-908, 2000.

[73] B. Yu, and M. P. Singh; An evidential model of distributed reputation management; In Proc. of the First Int. Conference on AAMAS. ACM Press, pp. 294-301, 2002.

[74] J. Sabatar; Trust and reputation for agent societies; Phd thesis, Universitat autonoma de Barcelona, 2003.

[75] Y. Wang, and M.P. Singh; Formal trust model for multiagent ststems; Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI), pp. 1551-1556, 2007.

[76] P.B. Velloso, R.P. Laufer, M.B. Duarte and G. Pujolle; HIT: A humaninspired trust model; IFIP International Federation for Information Processing, vol 211, pp. 35-46, 2006.

[77] W. Song, V.V. Phoha and X. Xu; An adaptive recommendation trust model in multiagent system; Intelligent Agent Technology, 2004. (IAT 2004). Proceedings. IEEE/WIC/ACM, pp. 462- 465, 2004.

[78] J. Shi, G.V. Bochmann and C. Adams; A trust model with statistical foundation; IFIP International Federation for Information Processing, vol 173, pp.145-158, 2005.

[79] J.V. Neumann and O. Morgenstern; Theory of Games and Economic Behavior; Princeton University Press 1944.

[80] M.J. Osborne and A. Rubinstein; A course in game theory; MIT Press, Cambridge, MA (1994).

[81] G. Zlotkin and J. S. Rosenschein; Coalition, cryptography and stability: Mechanisms for coalition formation in task oriented domains; In Proc. of AAAI94

[82] Sandhol; emediator: a next generation electronic commerce server; Computational Intelligence. v18 i4. 656-676.

[83] H.S. Nwana; Software agents: an overview; The Knowledge Engineering Review, 11(3), 1996, pp 205-244.

[84] P. Faratin, C. Sierra and N. R. Jennings; Using Similarity Criteria to Make Negotiation Trade-Offs; ICMAS 2000: 119-126

[85] S. Kraus; Strategic Negotiation in Multiagent Environments; MIT Press, 2001; ISBN: 0-262-11264-7

[86] N.R. Jennings; An agent-based approach for building complex software systems. Communications of the ACM 44(4), 35-41, 2001.

[87] Y. Labrou, T. Finin and Y. Peng; Agent communication languages: the current landscape; IEEE Intelligent Systems (1999), pp. 45–52.

[88] J. Searle; Speech Acts: An Essay in the Philosophy of Language; Cambridge University Press, 1969.

[89] D. R. Traum; Speech acts for dialogue agents; Foundations of Rational Agency , 1999.

[90] J. May, Y. Labrou and T. Finin; Evaluating KQML as an Agent Communication Language; Intelligent Agents II (LNAI Volume 1037). Springer-Verlag, 1996.

[91] http://www.fipa.org/specs/fipa00086/index.html

Appendix 1: Agent Definition File (ADF)

```
<?xml version="1.0" encoding="UTF-8"?>
<!--This is for Trust project.-->
<agent xmlns="http://jadex.sourceforge.net/jadex"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://jadex.sourceforge.net/jadex
                    http://jadex.sourceforge.net/jadex-0.96.xsd"
        name="Cosumer"
        package="Consumer">

        <imports>
                <import>java.util.*</import>
                <import>jadex.adapter.fipa.*</import>
          <import>jadex.planlib.*</import>
        </imports>

        <capabilities>
          <capability name="procap" file="jadex.planlib.Protocols"/>
                <!-- Include the directory facilitator capability under the name dfcap. -->
                <capability name="dfcap" file="jadex.planlib.DF"/>
        </capabilities>

        <beliefs>
                <!-- This belief contains the agents in the whole environment as an array. -->
                <belief name="names" class="String[]">
                <fact>new String[] </fact>

          </belief>

           <belief name="circle" class="Object">
             <fact> new ArrayList() </fact>
          </belief>

          <belief name="mine" class="Object">
             <fact> new ArrayList() </fact>
          </belief>

           <belief name="known_Tr" class="Object">
             <fact> new Hashtable() </fact>
          </belief>

          <belief name="known_TR" class="Object">
             <fact> new Hashtable() </fact>
          </belief>

          <belief name="known_N" class="Object">
             <fact> new Hashtable() </fact>
        </belief>

        <belief name="utility" class="Integer">
          <fact> 0 </fact>
        </belief>

      </beliefs>

    <goals>
        <maintaingoalref name="df_keep_registered">
                        <concrete ref="dfcap.df_keep_registered"/>
              </maintaingoalref>

              <achievegoalref name="rp_initiate">
                        <concrete ref="procap.rp_initiate"/>
              </achievegoalref>

              <!-- Include df search goal type from dfcap. -->
              <achievegoalref name="df_search">
                        <concrete ref="dfcap.df_search"/>
              </achievegoalref>
   </goals>

      <plans>
              <plan name="select">
                        <body class="EvaluationPlan"/>
                           <trigger>
                                <messageevent ref="request_selection"/>
                        </trigger>
              </plan>
              <plan name="tell">
                        <body class="Mine_InformPlan"/>
                        <trigger>
                                <messageevent ref="request_tell"/>
                        </trigger>
              </plan>
        <plan name="give">
                        <body class="Value_InformPlan"/>
                        <trigger>
                                <messageevent ref="request_what"/>
                        </trigger>
              </plan>
        <plan name="final">
                        <body class="Utility_InformPlan"/>
                        <trigger>
                                <messageevent ref="request_final"/>
                        </trigger>
              </plan>
        <plan name="init">
                        <body class="InitialPlan"/>
                        <trigger>
                        <messageevent ref="request_init"/>
                </trigger>
        </plan>
    </plans>

      <events>
        <messageevent name="request_init" direction="receive" type="fipa">
                <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.REQUEST</value>
                </parameter>
                <parameter name="content-start" class="String" direction="fixed">
                        <value>"fire"</value>
                </parameter>
        </messageevent>

        <messageevent name="request_final" direction="receive" type="fipa">
                <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.REQUEST</value>
                </parameter>
                <parameter name="content-start" class="String" direction="fixed">
                        <value>"final"</value>
                </parameter>
        </messageevent>
        <messageevent name="request_tell" direction="receive" type="fipa">
                <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.REQUEST</value>
                </parameter>
                <parameter name="content-start" class="String" direction="fixed">
                        <value>"who"</value>
                </parameter>
        </messageevent>

        <messageevent name="request_selection" direction="receive" type="fipa">
                <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.REQUEST</value>
                </parameter>
                <parameter name="content-start" class="String" direction="fixed">
                        <value>"show"</value>
                </parameter>

        </messageevent>

        <messageevent name="request_what" direction="receive" type="fipa">
                <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.REQUEST</value>
                </parameter>
                <parameter name="content-start" class="String" direction="fixed">
                        <value>"referee value"</value>
                </parameter>
        </messageevent>
        <messageevent name="inform" direction="send" type="fipa">
```

```
                    <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.INFORM</value>
                    </parameter>

                </messageevent>
                <!-- The answer message after some error occurred. -->
                <messageevent name="failure" direction="send" type="fipa">
                    <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.FAILURE</value>
                    </parameter>
                </messageevent>
        </events>

    <properties>
                <!-- Only log outputs >= level are printed. -->
                <property name="logging.level">Level.INFO</property>
                <!-- The default parent handler prints out log messages on the console. -->
                <property name="logging.useParentHandlers">true</property>
                <!--<property name="debugging">true</property>-->
   </properties>

  <configurations>
                <configuration name="default">
            <goals>
                        <initialgoal ref="df_keep_registered">
                            <parameter ref="description">
                                <value>
                                    SFipa.createAgentDescription(null,

        SFipa.createServiceDescription("service_trust", "do some_task", "University of
Concordia"))
                                </value>
                            </parameter>
                            <parameter ref="leasetime">
                                <value>300000</value>
                            </parameter>
                        </initialgoal>
                 </goals>
                <plans>
                        <initialplan ref="init"/>
                    </plans>
            </configuration>
        </configurations>

  </agent>
```

Appendix 2: Proof of Propositions and Theorems

Proof of Proposition 2. Without loss of generality, let $a, a_1, \dots, a_n$ be arguments in a given argumentation framework such that $a_1, \dots, a_n$ are the only attackers of a and a is the only attacker of these arguments. According to Definition 5, the argument a is not acceptable since it is attacked and not defended, directly or indirectly by a non-attacked argument. Because it is defended, a belongs to some preferred extensions. However, a does not belong to all of them. For example, a does not belong to the preferred extension to which the arguments $a_1, \dots, a_n$ belong since these arguments belong also to some preferred extensions because they are defended. a is then semi-acceptable.

Proof of Proposition 3. We prove this proposition by a counter example using Example **Error! Reference source not found.**. In this example $\{a, d\}$ and $\{b, d\}$ are complete extensions (preferred extensions). However, $\{d\}$ is not a complete extension.

Proof of Proposition 4. By Definition 5, the grounded extension is included in all preferred extensions. Consequently, using definition 4, an eliminated argument is not acceptable. Also, according to Definition 7, an eliminated argument is not semi-acceptable and not preferred semi-acceptable.

Proof of Proposition 5. Suppose that $\exists x \in \mathcal{WF}: a_x^p$ is not acceptable. Therefore, a part of the non-missing part of a_x^p is not acceptable. Because this part is also a part of a, then a is not acceptable. Contradiction!

Proof of Proposition 8. To prove this we should prove that $C_{as3} \Rightarrow \neg C_{as1} \wedge \neg C_{as2}$. Using the logical calculation, we can easily prove that $\neg C_{as1} \wedge \neg C_{as2} = \neg C_{as1} \wedge \neg Op_{as_2}^{qu_1} \wedge \neg Op_{as_2}^{qu_2}$. Also, if an agent β can build an acceptable argument a from $\mathcal{A}_\beta \cup CS_\alpha$, then it cannot build an acceptable or (preferred) semi-acceptable argument attacking a from the same set. Therefore, $\mathcal{AR}(\mathcal{A}_\beta \cup CS_\alpha) \rhd a \Rightarrow \neg C_{as1}$. Thus the result follows.

Proof of Theorem 1. Agents' knowledge bases are finite and repeating moves with the same content is prohibited. Consequently, the number of $Attack$ and $Question$ moves that agents can play is finite. At a given moment, agents will have two possibilities only: $Accept$ if an acceptable argument can be built from $CS_\alpha \cup CS_\beta$, or $Stop$, otherwise. Therefore, the protocol terminates successfully by $Accept$, or unsuccessfully by $Stop$ when $Accept$ move cannot be played, which means that only semi-acceptable arguments are included in $CS_\alpha \cup CS_\beta$.

Proof of Theorem 2. For simplicity and without loss of generality, we suppose that agent α starts the persuasion.

Let us first prove the $\Rightarrow$ direction: $\mathcal{AR}(\mathcal{A}_\alpha \cup \mathcal{A}_\beta) \rhd a \Rightarrow \mathcal{AR}(\cup CS) \rhd a$.

In the protocol, the persuasion starts when a conflict over p occurs. Consequently, the case where $\mathcal{A}_\alpha \rhd a$ and $\mathcal{A}_\beta \rhd a$ does not hold. The possible cases are limited to three:

1 $\mathcal{A}_\alpha \rhd a$ and $\mathcal{A}_\beta \not\rhd a$. In this case, agent α starts the persuasion over p by asserting a. Agent β can either play the $Attack$ move or the $Question$ move. Because $\mathcal{AR}(\mathcal{A}_\alpha \cup \mathcal{A}_\beta \rhd a)$ all the β's arguments will be counter-attacked. For the same reason, β cannot play the $Stop$ move. Consequently, at the end, β will play an $Accept$ move. It follows that $\mathcal{AR}(\cup CS \rhd a)$.

2 $\mathcal{A}_\alpha \not\rhd a$ and $\mathcal{A}_\beta \rhd a$. In this case, agent α starts the persuasion by asserting an acceptable argument b in its knowledge base against p

$(\mathcal{A}_\alpha \rhd b)$. This argument will be attacked by agent β, and the rest is identical to case 1 by substituting agent roles.

3 $\mathcal{A}_\alpha \not\rhd a$ and $\mathcal{A}_\beta \not\rhd a$. To construct argument a out of $\mathcal{A}_\alpha \cup \mathcal{A}_\beta$, two cases are possible. Either, (1) agent α has an acceptable partial argument a_∂^γ for p and agent β has the missing assumptions (or some parts of the missing assumptions, and agent α has the other parts), or (2) the opposite (i.e., agent β has an acceptable partial argument a_∂^γ for p and agent α has the missing assumptions (or some parts of the missing assumptions, and agent β has the other parts)). Only the second case is possible since the first one is excluded by hypothesis. For simplicity, we suppose that agent α has all the missing assumptions, otherwise the missing assumptions will be built by exchanging the different partial arguments. Agent α starts the persuasion by asserting an acceptable argument b in its knowledge base against p. Agent β can either play an $Attack$ or a $Question$ move. If attack is possible, then agent α can either counter-attack or play the $Stop$ move. The same scenario continues until agent α plays $Stop$, and then agent β plays a $Question$ Move. Agent α answers now the question by providing the missing assumptions, after which agent β attacks and agent α can only accept since $\mathcal{AR}(\mathcal{A}_\alpha \cup \mathcal{A}_\beta \rhd a)$. It follows that $\mathcal{AR}(\cup\, CS \rhd a)$.

Let us now prove the $\Leftarrow$ direction: $\mathcal{AR}(\cup\, CS) \rhd a \Rightarrow \mathcal{AR}(\mathcal{A}_\alpha \cup \mathcal{A}_\beta) \rhd a$.

In the protocol, to have $\mathcal{AR}(\cup\, CS) \rhd a$ one of the two agents, say agent α, puts forward the argument a and the other, agent β, accepts it. On the one hand, to advance an argument, agent

α plays the $Assert$ move (in the initial or question rules) or $Attack$ move (in the assertion or attack rules). In all these cases, we have: $\mathcal{AR}(\mathcal{A}_\alpha \cup CS_\beta) \rhd a$ and there is no partial acceptable argument attacking a from $\mathcal{A}_\alpha \cup CS_\beta$. On the other hand, to accept an argument (in the assertion or attack rules), agent β should check that $\mathcal{AR}(\mathcal{A}_\beta \cup CS_\alpha) \rhd a$, there is no other arguments changing the status of the persuasion topic, and there is no partial acceptable argument attacking a from $\mathcal{A}_\beta \cup CS_\alpha$. Therefore we obtain: $\mathcal{AR}(\mathcal{A}_\alpha \cup CS_\beta \cup \mathcal{A}_\beta \cup CS_\alpha) \rhd a$. Because $CS_\alpha \subseteq \mathcal{A}_\alpha$ and $CS_\beta \subseteq \mathcal{A}_\beta$ we are done.