Dialogue Games and Trust for Communicating Agents

Jihad Labban

A thesis

In

The Department

Of

Concordia Institute of 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

Your file Votre référence
ISBN: 978-0-494-40930-5
Our file Notre référence
ISBN: 978-0-494-40930-5

Abstract

Dialogue Games and Trust for Communicating Agents

Jihad Labban

Multi-agent applications are primarily based on agent interactions, which are constrained by the trust of participating agents. Two important issues in these applications are how agents can communicate in a flexible and efficient way and how an agent can authenticate information conveyed by other agents in the system. In this thesis, we present a new communication framework and trust model addressing these issues by considering three factors. The first factor is about the flexibility, complexity, soundness, and completeness of the communication protocol. The second factor is about the classification of agents from a trust point of view using direct interactions. The third factor is related to the categorization of the agent's chains through which the information is transmitted. Such a categorization is based upon the reliability of the agents in the chain. The model aims to examine all available data in order to determine the trustworthiness of agents as transmitters of information. This approach is the first attempt in multi-agent systems towards classifying agents in order to accomplish trust. We also propose a thorough set of criteria and policies to assign different degrees of trustworthiness to each agent and consequently to the chains in which they appear. Agents are considered autonomous and they interact flexibly using a set of logical rules called dialogue games. Termination, soundness, and completeness results of the communication protocol are proven and its computational complexity is addressed. The proposed approach is also evaluated.

Keywords: Trust, Dialogue Games, Multi-Agent Systems, Agent Types, Agent Characteristics, Chain of Agents.

Acknowledgments

I would like to take this opportunity to acknowledge all those who helped me during this thesis work. I would like to thank my supervisor Dr. Jamal Bentahar for introducing me to the world of multi-agent systems, his valuable suggestions and guidance during the course of this thesis work, as well as his patience in reviewing the manuscript.

I would also like to thank all the faculty and staff at the Concordia Institute for Information Systems Engineering at Concordia University for their assistance during my master's course-work. As for my fellow graduate students who offered great help during my study, I would like to thank them all. I would also like to thank Ms Wei Wan for her precious help in the implementation of the proposed model. Finally yet importantly, I would like to thank my parents and wife for their endless support and encouragement.

Table of contents

List of Figures

List of Tables

Chapter 1: Introduction

This chapter introduces the research context, which deals with agent interactions and the constraint related to these interactions, specifically the constraint of trust. Motivations, technological challenges, objectives and contributions of the thesis are also presented.

1.1 Context of the Research

In recent years, multi-agent systems have attracted the attention of researchers from many disciplines, arranging from computer science to philosophy to business. A multi-agent system is a system where agents are allowed to freely enter and leave the system, thus making the environment continuously changing. These agents have different resources and different skills and need to communicate with each other to collaboratively perform some tasks. As a result, these agent societies will become more and more similar to the human ones [1], and just as in real societies, these agent societies will need to interact and overcome the obstacles associated with these interactions. The major obstacle in such interactions is the trust aspect. The issue is how an agent in a multi-agent setting can flexibly and efficiently interact with other agents in the system and authenticate information conveyed by them.

Trust plays an important role in agent communication. The idea is how an agent can trust another when the environment is highly dynamic. In order to build a virtual multi-agent society that is more and more similar to a human society, we cannot assume that agents behave similarly. Instead, just like in the human societies there are good, bad, and malicious agents.

In this thesis, we will focus on the ability of agents to interact and on the reliability of these interacting agents within a multi-agent system. We are interested in the communication mechanisms and in the properties of the agents involved in the interactions. The issue is to design an interaction framework and check its soundness, as well as identify the agents' properties that will allow the agent community to decide about the truthfulness of a given agent. This can be done by verifying if the agent fulfills some requirements. These agents can reason about trust and are equipped with interaction capabilities through a set of dialogue games they can play [15, 17]. In the proposed framework, dialogue games are specified as formal rules governing the interactions between agents by specifying the allowed communicative acts (or the legal moves) agents can perform in different situations.

There exist three types of interactions in agent-based computing:

Coordination: Coordination is a property of the multi-agent system performing some activity in a shared environment. Cooperation is coordination among non antagonistic agents.

Collaboration: Collaboration means working together, and often refers to forms of high-level cooperation that require a mutual understanding and a shared view of the task being solved by several interacting agents [2].

Negotiation: Negotiation is a process involving dealing among agents, which are intended to result in an agreement and commitment to a course of action.

Since the proposed model should be compatible with all forms of interactions, the general term of "interaction" will be used throughout this thesis.

1.2 Motivations

To be able to interact flexibly in dynamic environments, agents need to use advanced communication mechanisms and to achieve trust. The 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 accept and authenticate the information that is being transmitted through these agents.

Another motivation is to allow these agents to use their autonomous and reasoning characteristics through an argumentation system helping them to argue about their trust within the dialogue game-based communication framework.

1.3 Technological Challenges

Agent technology is growing with time particularly since Distributed Artificial Intelligence (DAI) was introduced. As this technology advances, more challenges arise. Some of these challenges are listed below:

1. Increase quality of agent systems to industrial standard.
2. Provide effective agreed standards to allow open systems development.
3. Provide flexible communication mechanisms for open agent communities.
4. Develop reasoning capabilities for agents in open environments.
5. Develop agent ability to adapt to changes in the environment.
6. Develop trust models for open agent societies.

In this thesis, interests are focused on the last five challenges, more specifically 3, 4, and 6. The thesis will tackle two of these challenges, the third one by introducing an argumentation-based model for agent communication using dialogue games (Chapter 3), and the sixth challenge will be addressed by introducing a new trust model based upon agents' categories (Chapter 4).

1.4 Objectives

The first objective in this thesis is to develop an agent communication mechanism in which agents can reason about their communicative acts and decide about next acts in a flexible way instead of using rigid protocols like those utilized in networking. The second objective is to achieve trust in any form of interaction between agents using the proposed communication mechanism. These two objectives are accomplished by defining an agent communication framework based on argumentative dialogue games and by distinguishing two factors that influence trust. The first factor is how truthful is the information being transmitted, and the second factor is how trustworthy are the agents transmitting this information. The proposed communication mechanism is flexible because it is specified as a set of small logical rules that can be combined in different ways, and not as a non-decomposable entity that agents should execute.

The second objective is achieved by using two techniques, the first is based on the classification of agents from a trust point of view using direct interactions. The second is about the categorization of the agents' chains through which the information is transmitted. Such a categorization is based upon the reliability of the agents in the chain. The transmitted information in these interactions is composed of two parts: The information itself and the chain of agents by which the information is obtained [37].

The model assumes that information may seem to be accurate and reasonable, but to be accepted it needs an authentic chain with reliable agents. A chain represents the link between

different agents within a multi-agent system. This link is a result of direct interaction among agents in the chain. For example $Ag1$ has interacted with $Ag2$, and $Ag2$ with $Ag3$ and so on until the information is obtained. The classification of agents uses a rating system to classify the agents' trustworthiness. The ratings are a result of successful direct interactions.

Many interesting results in the proposed communication framework such as termination, soundness, completeness and complexity analysis are proved. This framework along with the trust model is evaluated through a prototype implemented in agent-based programming using Jadex [41].

1.5 Contributions

The thesis contributions are summarized in two points:

1. Introducing an approach for agent communication based on argumentative dialogue games.
2. Proposing a new trust model based on agent Categorization.

1.6 Thesis Overview

The rest of the thesis is organized as follows. Chapter 2 presents the state-of-the-art. In this chapter, the concepts of agents and multi-agent system are introduced. Agent communication and trust in agent societies are also presented. Chapter 3 presents the first contribution: an agent communication framework based on argumentation and formal dialogue games. Chapter 4 discusses the proposed trust model based on agent categorization. Implementation issues are presented and discussed in Chapter 5. Chapter 6 concludes the thesis.

Chapter 2: Multi-Agent Systems: Communication and Trust considerations

2.1 Introduction

For years, researchers in artificial intelligence (AI) have tried to make computer resources 'learn', to simulate human intelligence processes by machines. These processes include learning, reasoning, and self-correction. Examples of where AI is used include speech-recognition technology. This is where software attempts to make decisions on the spelling of words based on the context in which they are used. Anyone who has used speech-recognition technologies in a business context will know that the result can be, at best, laughable. For example should one's voice be changed by something like a cold, the whole thing will fall apart. In other words such technologies do not always work as well as we would like. Modern technology however has advanced since (AI) was first developed. A multi-agent system can have:

1. Anywhere from one to thousands or millions of agents.
2. Heterogeneous or homogeneous agents.
3. Cooperating or competitive agents.
4. Simple or complex agents.
5. Simple or complex goals.
6. System and local goals that vary or that are the same across agents.

In order to describe what an intelligent agent is, and from that explain what a multi-agent system is and its usage, we have to examine the most powerful agent of all time-the human being. We should be looking at what is called the "genius" behind any computer. Indeed, computers are not very good at knowing what to do: every action a computer performs must be explicitly anticipated, planned for, and coded by a programmer. From this introduction, we have to credit

the programmer for the success of a computer or an agent. Figure 1, presents all the different factors that influence a multi-agent system.

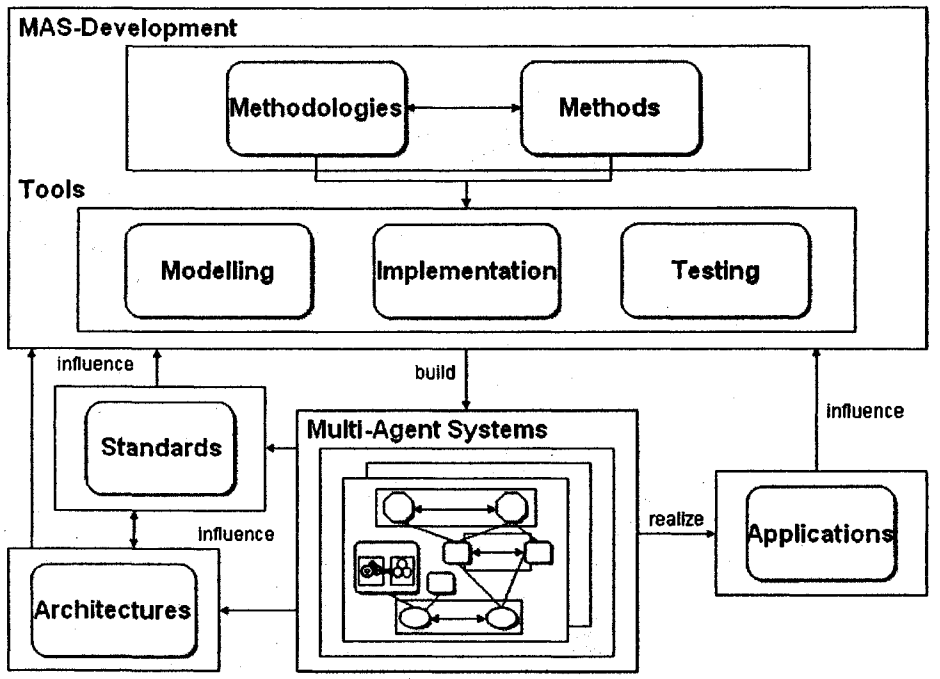


Figure 1 Development of Multi-Agent Systems

Methodologies are needed to systematically guide and support agent software engineers through the various stages of system development. They therefore form the framework and define the rules for the whole development process.

In each step of the development, the methodology proposes a certain method that can be used to define the necessary system artefacts. To be able to build industrial-strength agent oriented applications, sound methodologies, and methods are of great importance [41].

Methodologies and methods are the necessary technical foundation for building complex agent systems. Nevertheless, additional practical support is needed to enable a developer to utilize the concepts from the methodology. Therefore, a continuous *tool-*

support considering all stages of the development is needed. The design artefacts created at one step should be refined and reused in the following steps and should lead in a natural way to an executable system specification. Tool support is necessary in different phases of the development process. In the *modelling phase* an analysis and design model of the problem domain should be produced. In the *implementation phase* the design model is extended to executable code that runs on a multi-agent platform. The multi-agent platforms decide on the transfer possibilities and dictate the MAS-internal and social architectures that can be used. For a smooth transition, it is desirable to have the same bundle of concepts at the design and implementation level. To enable the system debugging, tool support is also necessary at the *testing level.* The various design artefacts represent different views of the whole system and can be utilized to establish tools that emphasize varying multi-agent aspects. Concrete multi-agent systems are built to realize (components of) applications. Different *application classes* exist, ranging from enterprise information systems to computer games, which highlight different aspects of the agent paradigm (e.g. autonomy), and require different types of agents. On a more abstract level, one can try to extract the general *success factors* for agent technology to provide generic criteria, allowing selecting suitable agent-based solutions to problems of specific application domains [41]. In Figure 2, the components of multi-agent systems are shown.

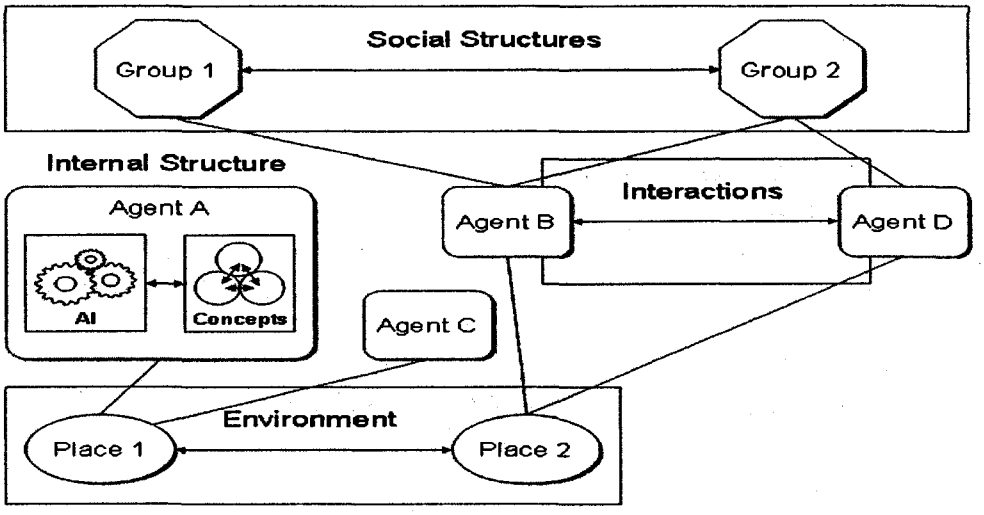


Figure 2 Structure of Multi-Agent Systems

Agents are social entities in the sense that they are often situated in an environment with other agents. In order to fulfill its goals, an agent frequently has to delegate tasks to other agents, because it has not all the needed capabilities to reach the goal solely by itself. Therefore, the social structures in which an agent resides are important for the workflows in the completely multi-agent system. Agents are social beings in the sense that they have the possibility to initiate interactions with other agents. Research in the field of agent interactions covers the different aspects of communication at varying abstraction levels [41].

Fields of interest are the agent communication foundations, conversations, and cooperation. Agent communication takes place on a more abstract level compared to the case with object-oriented communication. Agent communication is based on the speech act theory [42]. Communication is treated as a way of acting, as certain kinds of natural language utterances have the characteristics of actions (called *speech acts*). S*peech acts* will be discussed in Chapter 3.

2.2 What are Agents?

An agent is a software program capable of flexible, autonomous (problem-solving) action, and situated in dynamic, open, and unpredictable environments. An agent has control over its internal state and over its own behaviour. It experiences environment through sensors and acts through effectors. These software programs communicate with each other and act in a multi-agent setting to resolve some problems such as matching available resources to demand.

Agents are viewed in the multi-agent society as a metaphor for the design of complex and distributed computational systems. The agent community is interested in the following aspects: multi-agent planning, agent communication languages, coordination mechanisms, matchmaking architectures, agent programming languages, auctions, negotiation, mechanism design, strategies, and learning.

Weiss [2] 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 behaviour that is a consequence of its observations, its knowledge and its interactions with the other agents*". Fig 3 represents the other characteristics of an agent.

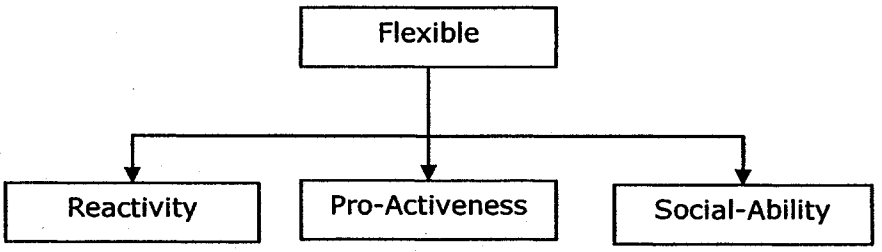


Figure 3 Agent characteristics

Reactivity means that the agent is able to perceive its environment, and to respond in a timely fashion to change, in order to satisfy designer's objectives.

Pro-activeness means that the agent exhibits goal directed behaviour, and is able to take initiative actions.

Social ability means that the agent is able to interact with other agents in order to satisfy designer's objectives.

Agent technology is used in many applications such as automatic negotiation, auctions, contract dealing from the supply chain management point of view, simulation, network management, e-business (e-commerce) and many more.

2.3 Agent Negotiation

Negotiation is an important type of interaction that is gaining increasing prominence in the agent community. Agents with conflicting interests, but with a desire to cooperate, try to come to a mutually acceptable agreement on the division of scarce resources [3, 4, 5, and 6]. Resources can be commodities, services, time, money, etc. Resources are scarce in the sense that competing claims over them cannot be fully satisfied simultaneously. The problem of resource negotiation via communication 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 [6, 12, 15, 17, 20, 25, and 26]. 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 [3, 9, 26, and 40].

Agent communication is related to several disciplines: philosophy of language, social psychology, artificial intelligence, logic, mathematics, etc. In order to be able to negotiate, solve conflicts of interest, cooperate, find proofs, agents need not only exchange single messages, but also take part in conversations with other agents. A conversation is defined as a coherent

sequence of utterances. The term "coherent" means that the information conveyed by an utterance is related to the information conveyed by the other utterances in a conversation. For example, if p is the information conveyed by an utterance, 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.

In the recent research into agent negotiation, flexible protocols based on dialogue games are used [3, 9, and 40]. Specifying dialogue games using argumentation is being used to overcome some of the negotiation issues. Simply put, argumentation is the process of building arguments for or against some conclusion. Chapter 3 presents an argumentation-based dialogue game protocol and shows how it contributes to the negotiation issues.

2.4 Securing Agent Negotiation

In multi-agent systems, negotiation takes place between two or more software programs. The demand for security in such a context is increasing, particularly because multi-agent systems are open in nature. Unlike network security issues, where digital credentials (e.g. credit, driver licence, membership number etc...) can be used, in multi-agent systems, in order to secure a negotiation involving software programs, one has to first make sure the agents trust each other and then one will carry on the negotiation. The objective is to propose trust models in order to implement a secure setting for automatic negotiation. Two of these models have been proven to be efficient: the FIRE model and the Referral model. After defining the meaning of trust, these two models will be discussed in this chapter. The proposed trust model presented in Chapter 4 uses some ideas from these two models.

2.5 What is the Meaning of Trust?

Trust is generally considered a nebulous term to define, but everyone has a threshold or set of circumstances when he trusts some entity as well as when he does not.

The following definitions provide a useful starting point for defining trust:

- IETF: "Generally, an entity can be said to 'trust' a second entity when it (the first entity) makes the assumption that the second entity will behave exactly as the first entity expects. This trust may apply only for some specific function [33].
- Rob Brickman from IBM asserts that "trust can be defined as assured reliance on the character, ability, strength, or truth of someone or something [34].

2.6 The FIRE Model

FIRE [35] is a trust and reputation model that integrates a number of information sources to produce a comprehensive assessment of an agent's likely performance. FIRE incorporates:

Interaction trust (IT): models the trust that ensues from the direct interactions between two agents. In more detail, consider a commercial transaction where agent a buys a particular product from agent *b*. The outcome of the transaction may consist of the product price, product quality, and the delivery date. From this outcome, agent a may give ratings about agent *b*'s service in terms of price, quality, and delivery for that particular interaction.

Ratings are thus tuples in the following form: r = (*a; b*; *i*; *c*; *v*); where *a* and *b* are the agents that participated in the interaction *i*, and *v* is the rating *a* gave to *b* for the term *c* (e.g. price, quality, delivery). The range of *v* is [-1; +1], where -1 means absolutely negative, +1 means absolutely positive, and 0 means neutral or uncertain.

In order to calculate IT from past experiences, an agent needs to record its past ratings in a (local) rating database. When calculating the IT value for agent *b* with respect to term *c*, agent *a*

has to query its database for all the ratings that have the form (*a, b, c, _, _*) where the '_' symbol can be replaced by any value. The set of those ratings is denoted by *R*(*a, b, c*). Then the IT (denoted by Γ_i) is calculated as the weighted mean of the rating values of all the ratings in the set:

$$\Gamma_i(a,b,c) = \sum_{r_i \in R(a,b,c)} w(r_i) \cdot v_i \quad (1)$$

Where v_i is the value of the rating r_i and $w(r_i)$ is the weight corresponding to r_i. The weight $w(r_i)$ for each rating is selected such that it gives more weight to more recent ratings, with a constraint that $\sum_{r_i \in R(a,b,c)} w(r_i) = 1$. This is to ensure that the trust value $\Gamma_i(a,b,c)$ is in the range [-1; +1].

In FIRE, each trust value comes with a reliability rating that reflects the confidence of the trust model in producing that trust value given the data it took into account. This value is built from the two following measures:

$\rho_N(a,b,c)$: Is the reliability measure based on the number of ratings that have been taken into account in computing Γ_i. As the number of these ratings (*n*) grows, the degree of reliability increases until it reaches a defined threshold (denoted by m).

$$\rho_N(a,b,c) = \begin{cases} n/m & when\ n \le m \\ 1 & when\ n > m \end{cases} \quad (2)$$

Where *n* is the cardinality of the set *R*(*a, b, c*). The value of function n/m ranges from 0 to 1 for *n* in [0; m]. Hence, the reliability $\rho_N(a,b,c)$ increases from 0 to 1 when the number of ratings *n* increases from 0 to m, and stays at 1 when n exceeds *m*.

$\rho_D(a,b,c)$: is the rating deviation reliability. The greater the variability in the rating values, the more volatile the other agent will be in fulfilling its agreements:

$$\rho_D(a,b,c) = 1 - \sum_{r_i \in R(a,b,c)} (w(r_i) \cdot |v_i - T_i(a,b,c)|)/2 \quad (3)$$

Then, the reliability measure of IT (called ½TI (*a; b; c*)) is defined by the following formula:

$$\rho_{\Gamma(a,b,c)} = \rho_N \bullet \rho_D \quad (4)$$

Role-based trust (RT): models the trust resulting from the role-based relationships between two agents (e.g. owned by the same organization or relationships derived from relationships between the agents' owners in the real life such as friendship or relatives). Since there is no general method for computationally quantifying trust based on this type of relationship, the authors use rules to assign RT values. This means end users can add new rules to customize this component to suit their particular applications. Rules are tuples of the following form:

$rul = (role_a, role_b, c, v_D, e_D)$, which describes a rule that if $role_a$ and $role_b$ are the roles of agent a and *b* respectively, then the expected performance of b in an interaction with a is v_D ($v_D \in [-1; 1]$) with respect to the term c; $e_D \in [0;1]$ is the default level of influence of this rule on the resulting RT value. For example, possible rules may be:

rul_1 = (buyer; seller; quality;-0.2; 0:3),

rul_2 = (friend-buyer, friend-seller, quality,0, 0.6), and

rul_3 = (-, government-seller, quality, 0, 0.9).

rul_1 expresses an agent's belief that an ordinary seller will usually sell a product of slightly lower quality than agreed, but the reliability of this belief is low (0:3); rul_2 is the belief that in a close partnership the buying agent can expect the seller to do what is agreed in terms of product quality; and this is also true for a governmental seller almost all of the time (rul_3).

Each agent has its own set of rules which are stored in a (local) rule database. In order to determine the RT with an agent b, agent a looks up the relevant rules from its rule database. Then the value of RT is given by the following formula:

$$\Gamma_R(a,b,c) = \frac{\sum_{rul_i \in Rules(a,b,c)} e_{Di} \cdot v_{Di}}{\sum_{rul_i \in Rules(a,b,c)} e_{Di}} \quad (5)$$

Where rul_i = ($role_a$; $role_b$; c; v_{Di}; e_{Di}) is a rule in the set of Rules (*a; b; c*). This set is a subset of the rule database in which only the rules that are relevant to the roles of a, the roles of b, and the term c are selected. Since the rules for RT are specified by the agent's owner, the reliability of RT also needs to be set by the agent's owner. The authors use $\rho_{\Gamma_r}(a,b,c)$, with a range of [0, 1], to denote this value.

Witness reputation The witness reputation (WR) of a target agent *b* is built on observations about its behaviour by other agents (witnesses). In order to evaluate the (WR) of agent *b*, agent *a* needs to find the witnesses that has interacted with agent *b*.

In this component, a variant of the referral system is used to find such witnesses. In this system, agents cooperate by giving, pursuing, and evaluating referrals (a recommendation to contact another agent). Each agent in the system maintains a list of acquaintances (other agents that it knows). Thus, when looking for a certain piece of information, an agent can send the query to a number of its acquaintances who will try to answer the query if possible or, if they cannot, they will send back referrals pointing to other agents that they believe are likely to have the desired information.

In this model, each agent has a measure of the degree of likeliness with which an agent can fulfill an information query. This measure needs to be defined in an application specific manner. An agent is assumed to know local agents (those who are near to it) better and so the distance between an acquaintance and the target agent is used as the knowledge measure. Thus the nearer to the target agent, the more likely the acquaintance is to know it. When an agent *a* assesses the WR of an agent *b* with respect to a term *c*, denoted by $\Gamma_W(a,b,c)$, it sends out a query for ratings of the form (-,*b*,-,*c*,-), to those acquaintances that are likely to have relevant ratings on agent *b* and term *c*. These acquaintances, upon receiving the query, try to match it to their own rating databases. If they find matching ratings, it means they have had interactions with agent *b*, and they will return the ratings found to agent *a*. If they cannot find the requested information,

they will return referrals identifying their acquaintances that they believe are most likely to have the relevant ratings to the query so that agent *a* can look further.

This process continues until sufficient witnesses are found or the lengths of its referral chains reach a defined threshold (because the further the witness is from agent *a*, the less reliable/relevant its information is to it). The general formula for WR is as follows:

$$\Gamma_W(a,b,c) = \sum_{r_i} w(r_i).v_i \quad (6)$$

Where $\Gamma_W(a,b,c)$ is the set of witness ratings found by agent *a*, $w(r_i)$ is the weight for each rating and v_i is the rating value of r_i. The reliability measure for WR (denoted by $\rho_{\Gamma_W}(a,b,c)$) is also defined from the ratings in $\Gamma_W(a,b,c)$.

Certified reputation (CR): is based on ratings presented by the rated agent (*agent b*) about itself which have been obtained from its partners in past interactions. These ratings are certifications (provided by the rating agents) of agent *b's* past performance (somewhat like a reference when applying for a job). They allow an agent to prove its achievable performance as viewed by previous interaction partners.

Since agent *b* can choose which ratings it puts forward, a rational agent will only present its best ratings. Therefore, we should assume that (CR) information possibly overestimates an agent's expected behaviour. Thus, although it cannot guarantee agent *b's* performance in future interactions, the (CR) information does reveal a partial perspective on agent *b's* past behaviour. The main benefit of this type of information is its high availability. With the cooperation of its partners, agent *b* can have (CR) information from just a small number of interactions. Therefore, (CR) is available to agents in most circumstances; even in situations where the other components may fail to provide a trust measure. In more detail, the process of (CR) is as follows:

- After every transaction, agent *b* asks its partners to provide their ratings about its performance, which is then stored in its databases.

- When agent a contacts agent b to express its interest in using b's services; it asks agent b to provide references about its past performance.
- As agent a receives the ratings of agent b from b. It assesses the ratings' reliability and calculates a trust value for agent b. Specifically, the value of (CR), $\Gamma_C(a,b,c)$, and its reliability, $\rho_{\Gamma_C}(a,b,c)$, are calculated as per the (WR) component, however the input is the set of ratings provided by the target agent b itself.

The authors combine the aforementioned trust/reputation values into a single composite measure to give an overall picture of an agent's likely performance. Specifically, the weighted mean method is used to calculate the composite trust value ($\Gamma(a,b,c)$) and its reliability ($\rho_\Gamma(a,b,c)$):

$$\Gamma(a,b,c) = \frac{\sum_{k\in\{I,R,W,C\}} w_k.\Gamma_k(a,b,c)}{\sum_{k\in\{I,R,W,C\}} w_k} \quad (7)$$

$$\rho_\Gamma(a,b,c) = \frac{\sum_{k\in\{I,R,W,C\}} w_k}{\sum_{k\in\{I,R,W,C\}} W_k} \quad (8)$$

Where $w_k = W_k.\rho_k(a,b,c)$ and W_I, W_R, W_W, W_C are the coefficients corresponding to the IT, RT, WR, and CR components. These coefficients are set by end users to reflect the importance of each component in a particular application.

2.7 The Referral Model

This model [36] considers a distributed system of software agents who cooperate in helping their users to find services provided by different agents. The agents need to ensure that the service providers they select are trustworthy. Because we are dealing with autonomous agents and there is no central trusted authority, the agents should help each other to determine the trustworthiness of the service providers they are interested in. This help is rendered via a series of ***Referrals*** to other agents, culminating in zero or more trustworthy service providers being

identified. In order for agents to rate the trustworthiness of others, the referral model need to identify a "***Trust Network***" which is a multi-agent system.

The referral model needs to define two operators, ***concatenation,*** and ***aggregation*** for which trust ratings can be combined in a trust network. The referral system will try to establish some important properties regarding these operators, thereby ensuring that trust can be combined correctly.

Due to the uncertainty about quality and reliability of the product and services offered by others in open multi-agent systems, it is crucial for agents to compute the trustworthiness of the other agents before initiating any service request. Agents should be able to compute how much trust to place in others with which they might have had no prior experience. The mechanism that supports these findings on trust is what is known as the *Reputation systems.*

Collaborative filtering approaches select resources based on the relationships between agents by the similarities or dissimilarity in their subjective judgments'.

The Referral system tries to maintain and use trust for recommendation, that is trust is captured by the neighbourhood relation. This system uses a probabilistic theory to represent the trust between agents which is based on referrals and the quality of service obtained. It will attempt to use the operators mentioned above to determine trust between agents. Some of the challenges that the referral system must take into consideration for an open environment are:

1. Agents may join and leave the environment arbitrary.
2. The agents might not be cooperative.
3. An agent may give biased information on itself.
4. Agents might refer witnesses that are biased.

2.8 Conclusion

In this chapter, agent-based computing from communication and trust points of view has been presented. Agents are defined as autonomous and flexible entities, which form societies that depend on mechanisms governing their interactions in order to achieve trust among them. Agent negotiation is discussed and two trust models are introduced: the FIRE model and the Referral model. The FIRE model is based on four types of information related to trust and reputation (Interaction Trust, Role-based Trust, Witness Reputation, and certified Reputation). From this information, trust and reputation of interacting agents are measured. The second model is based on references from other agents when asked about a target agent. This model has the limit of assuming that agents will tell the truth. This assumption defies the fact that there are malicious agents in the environment.

Chapter 3: Multi-Agent Interactions using Formal Dialogue Games

3.1 Introduction

Developers of software agent systems typically design the agents within the system to perform changes in the state of the world. Whether the software agents represent human bidders in an online auction or the system collectively manages some resource, (such as a utility network), the agents usually need to initiate, maintain, or terminate actions in the world [7]. Agent interaction protocols, therefore, must be concerned with argument over actions: even if agents in such systems are not concerned with sharing and reconciling one another's beliefs, these protocols will still assist in sharing and coordinating their actions.

An important class of interactions between agents in multi-agent systems takes the form of dialogues. There is a great variety of dialogues ranging from exchange of pre-formatted messages to argumentation based dialogues.

Walton and Krabbe distinguish five types of dialogues [8] listed below:

Information seeking dialogue: are those where one agent seeks the answer to some question(s) from another agent.

Inquiry dialogue: the agents collaborate to answer some question(s).

Persuasion dialogue: involve one agent seeking to persuade another to accept a proposition.

Negotiation dialogue: the agents bargain over the division of some scarce resource.

Deliberation dialogue: agents collaborate to decide what action or course of actions should be adopted in some situation.

Formal dialogue games are being used by agent designers for structured agent interactions. They are the basis of interaction between autonomous software agents, where

each agent moves by making utterances, according to a pre-defined set of rules [9]. A dialogue game specification then consists of the following elements [10]:

Commencement Rules: are rules which define the circumstances under which a dialogue commences.

Locutions: are rules which indicate what utterances are permitted.

Combination Rules: are rules which define the dialogical contexts.

Commitments: are rules which define the circumstances under which agents express commitment to a proposition.

Termination Rules: are rules that define the circumstances under which a dialogue ends.

Various dialogue protocols can be found in the literature, especially for persuasion [11, 12, 15, 17], and negotiation [13, 14]. Multi-agent negotiation is based on two aspects, the first is the way the negotiation is carried out, and the second focuses on trust issues. Formal dialogue games is used to help with the negotiation procedure, (e.g. sending an offer, receiving counter offers, etc...); trust is another issue that is of concern in multi-agent systems.

Formal dialogue games are defined as a set of small games used to represent a negotiation, as if we have a big job to do, in order to accomplish it, we need to form a team and assign small tasks to each member, at the end we get the result of each task and combine them to get the final result. Negotiation is this big job, and formal dialogue games split the negotiation into small games and then compile them to get the result of the more complex negotiation.

3.2 Overview

In this chapter, the proposed dialogue game-based communication framework in a negotiation setting is presented. A formal description of this argumentation-driven negotiation protocol between autonomous agents is shown. This protocol is designed to be computationally efficient. Argumentation can be defined as a process for the interaction of different arguments for and against some conclusion [21, 22, and 23]. Argumentation has been researched extensively in

the last decade, especially for inference, decision support, dialogue, and negotiation. Agents can use argumentation in their communication in order to justify their negotiation stances and influence other agents' negotiation stances. An important branch of argumentation is formal dialectics [24, 25, 26, 21, 22, and 4]. In its most abstract form, a dialectical model is a set of arguments and a binary relation representing the attack-relation (and indirectly the defence relation) between the arguments in a dialogical process. Consequently, dialectical models are relevant for automated negotiation, in which agents should persuade each other. In this thesis, we propose to use dialectical argumentation to assist agents to reach a decision and convince each other.

A single agent may use such an argumentation to perform its reasoning because it needs to make decisions in highly dynamic environments, considering interacting preferences and utilities. This argumentation can also help multiple agents to interact rationally, by giving and receiving reasons for conclusions and decisions, within an enriching dialectical process that aims at reaching mutually agreeable joint decisions. During negotiation, agents can establish a common knowledge of each other's commitments, find compromises, and persuade one another to make commitments.

Several computational frameworks for argumentative inferences have been developed in the literature [24, 21, 3, 4, 23, 27]. However, only few proposals have considered the implementation and application issues of argumentation-based negotiation. Another challenging problem for automated negotiation that has not been deeply addressed is the security engineering.

The problem of securing negotiation systems in a distributed setting is of great utility for a number of applications such as e-commerce, e-business, and Web services-based applications. The objective of this chapter and the next one is to address the specification and security issues of agent-based negotiation. We propose a new computational model for efficient (in this chapter) and secure (in the next chapter) negotiation using an argumentation-driven framework and a trust-based approach. The idea is to be able to participate in flexible negotiations and to trust

interacting agents within a multi-agent system. This is because in order to share resources and allow mutual access, involved agents in e-infrastructures need to establish a framework of trust that establishes what they each expect of the other. Such a framework must allow one entity to assume that a second entity will behave exactly as the first entity expects.

3.3 Conceptual Framework

3.3.1 Theoretical Considerations

Agent communication is related to several disciplines: philosophy of language, social psychology, artificial intelligence, logic, mathematics, etc. In this domain, in order to be able to negotiate, solve conflicts of interest, cooperate, and find proofs, agents need not only exchange single messages, but also take part in conversations with other agents. A conversation is defined as a coherent sequence of utterances. The term "coherent" means that the information conveyed by an utterance is related to the information conveyed by the other utterances in a conversation. For example, if p is the information conveyed by an utterance, 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.

To consider the conversational aspect of agent communication, action logic is used to specify the communicative acts. In addition, to capture the formal semantics of such communicative acts, the approach is based on the notion of "social commitments" [25, 28, 29]. A social commitment (SC) is an engagement made by an agent (the debtor), that some fact is true or that something will be done. This commitment is directed to a set of agents (creditors). A commitment is an obligation in the sense that the debtor must respect and behave in accordance with this commitment. Social commitments are a powerful representation to model multi-agent interactions.

Commitments provide a basis for a normative framework that makes it possible to model agents' communicative behaviours. This framework has the advantage of being expressive because all speech act types can be represented by commitments [30]. Commitment-based protocols enable the content of agent interactions to be represented and reasoned about [29].

In order to model the dynamics of conversations, speech act (SA) is interpreted as an action performed on a commitment or on its content. A speech act is an abstract act that an agent, the speaker, performs when producing an utterance (U) and addressing it to another agent, the addressee. The game protocol for the negotiation dialogue introduced in section 3, the actions that an agent can perform on a commitment are: $Act \in \{Create, Withdraw\}$. Create means that the agent is making an offer, and Withdraw means that the agent is withdrawing it. The actions that an agent can perform on commitment content are: $Act\text{-}content \in \{Accept, Refuse, Challenge, Defend, Attack, and Justify\}$.

In the proposed framework, a speech act is interpreted either as an action applied to a commitment when the speaker is the debtor, or as an action applied to its content when the speaker is the debtor or the creditor [25]. Formally, a speech act can be defined as follows:

Definition 1 (Speech Act)

$$SA(Ag_1, Ag_2, U) \triangleq Act(Ag_1, SC(Ag_1, Ag_2, p)) \mid Act\text{-}content(Ag_k, SC(Ag_i, Ag_j, p))$$

Where $i, j \in \{1, 2\}$ and ($k = i$ or $k = j$), p is the commitment content.

The definiendum $SA(Ag_1, Ag_2, U)$ is defined by the definiens $Act(Ag_1, SC(Ag_1, Ag_2, p))$ as an action performed by the debtor Ag_1 on its commitment. The definiendum is defined by the definiens $Act\text{-}content(Ag_k, SC(Ag_i, Ag_j, p))$ as an action performed by an agent Ag_k (the debtor or the creditor) on the commitment content.

3.3.2 Architecture

The conceptual framework architecture of the model is characterized by capturing simultaneously 1) the mental aspect of agents taking part in the conversation (beliefs, desires, goals...), 2) the social aspect reflecting the context in which these agents communicate and the social commitments and norms, and 3) the reasoning aspect which is essential to be able to take part in coherent conversations. The reasoning part is based upon an argumentation system enabling agents to justify the facts on which they are committed and to justify their actions on commitments. The combination of these three aspects is necessary because producing social commitments (i.e. public utterances) reflects the agents' mental states on which agents should reason before committing in a conversation and during its unfolding.

The communication model consists of three layers: the conversation layer, the argumentative layer, and the cognitive layer. This stratification in layers is supported by the abstraction levels. The conversation layer is directly observable and highlights speech acts the agents perform. These acts are not performed in an isolated way, but within a particular conversation. The argumentative layer is used to correctly manage the social commitments and arguments that are related to the conversation. Finally, the cognitive layer is used to take into account the agents' private mental states, the social relations, and other elements that agents use to be able to communicate. In this paper we focus on the second layer.

In order to allow negotiating agents to use suitably the communication model, this latter must be compatible with the agent architecture. Thus, we propose a negotiating agent model consisting of a mental model, a social model, and a reasoning model (Figure 4). The mental model includes beliefs, desires, goals, etc. The social model captures the social concepts such as conventions, roles, commitments, etc. Commitments that agents make public by communication are related to the mental states via the reasoning model.

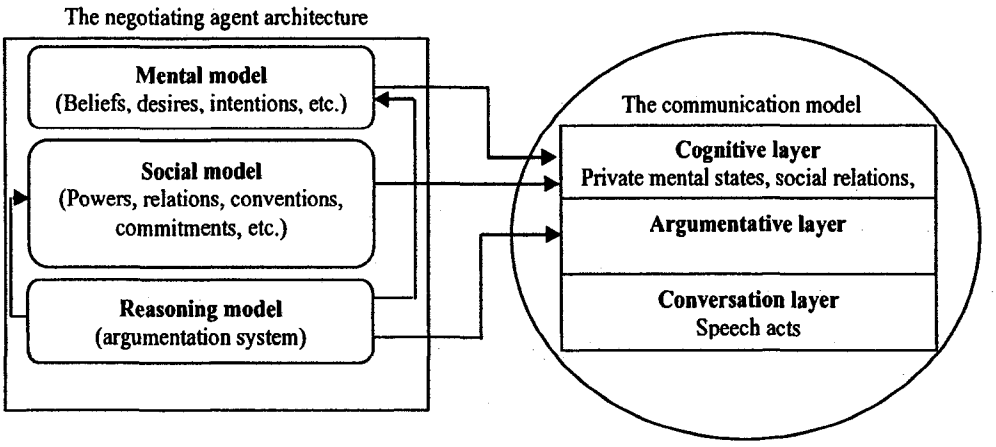


Figure 4 The conceptual framework

3.3.3 Argumentation Framework

The agents' reasoning model is specified using an argumentation system. Such a system essentially includes a logical language L, a definition of the *argument* concept, and a definition of the *attack* relation between arguments. The use of a logical language enables negotiating agents to use a logic-based reasoning in order to effectively reason about arguments in terms of inferring and justifying conclusions, and attacking and defending arguments. Hereafter the concepts that will be used in the negotiation are defined. Here Γ indicates a possibly inconsistent knowledge base with no deductive closure. "$\vdash$" Stands for classical inference and "$\equiv$" for logical equivalence.

Definition 2 (Argument) *An argument is a pair* (H, h) *where* h *is a formula of* L *and* H *a sub-set of* Γ *such that: i)* H *is consistent, ii)* $H \vdash h$ *and iii)* H *is minimal, so no subset of* H *satisfying both i and ii exists.* H *is called the support of the argument and* h *its conclusion. We use the notation:* $H = Support\ (Ag, h)$ *to indicate that agent* Ag *has a support* H *for the conclusion* h.

Definition 3 (Attack Relation) *Attack relation is a binary relation between two arguments.* Let (H_1, h_1) *and* (H_2, h_2) be two arguments, (H_1, h_1) attacks (H_2, h_2) is denoted: $(H_1,h_1) \not\Rightarrow (H_2,h_2)$.

$(H_1,h_1) \not\Rightarrow (H_2,h_2)$ *iff* $h_1 \vdash \neg h_2$.

In fact, before committing to some fact h being true (i.e. before making an offer by creating a commitment whose content is h); the speaker agent should use its argumentation system to build an argument (H, h). On the other side, the addressee agent must use its own argumentation system to select the answer it will give (i.e. to decide about the appropriate manipulation of the content of an existing commitment).

For example, an agent Ag_1 accepts the commitment content h proposed by another agent if Ag_1 has an argument for h. If Ag_1 has an argument neither for h, nor for $\neg h$, then it challenges h.

In the framework, we distinguish between arguments that an agent has (private arguments) and arguments that this agent used in its conversation (public arguments). Thus, the notation: $S = Create\_Support\ (Ag,\ SC\ (Ag_1, Ag_2, p))$ is used to indicate the set of commitments S created by agent Ag to support the content of $SC(Ag_1, Ag_2, p)$. This support relation is transitive i.e.:

$(SC\ (Ag_1, Ag_2, p_2) \in Create\_Support\ (Ag, SC\ (Ag_1, Ag_2, p_1))$

$\wedge\ SC\ (Ag_1, Ag_2, p_1) \in Create\_Support\ (Ag, SC\ (Ag_1, Ag_2, p_0)))$

$\Rightarrow SC(Ag_1, Ag_2, p_2) \in Create\_Support(Ag, SC(Ag_1, Ag_2, p_0))$

Surely, an argumentation system is essential to help agents justify their negotiation stances and influence other agents' negotiation stances. However, reasoning on other social attitudes should also be taken into account in order to explain the agents' decisions.

In the proposed approach, agents can reason about trust and use trustworthiness to decide, in some cases, about the acceptance of arguments. This trust-based reasoning is essential for securing negotiation settings.

3.3.4 Computational Dialogue Games

Agent communication protocols specify the rules of interaction governing a dialogue between autonomous agents in a multi-agent system. These protocols are patterns of behavior that restrict the range of allowed follow-up utterances at any stage during a dialogue. Unlike protocols used in distributed systems, agent communication protocols must take into account the fact that artificial agents are autonomous and proactive. These protocols must be flexible enough and must also be specified using a more expressive formalism than traditional formalisms such as finite state machines. Indeed, logic-based protocols seem an interesting way [26, 31].

A computational dialogue game [26, 3, 32] aims at offering more flexible protocols. This is achieved by combining different games to construct complete and more complex protocols. Dialogue games are declarative specifications that govern communication between autonomous agents. They are interactions between players, in which each player moves by performing utterances according to a pre-defined set of rules. In this thesis, we propose to formalize these games as a set of logical rules about which agents can reason in order to decide which game to play and how to combine games. Indeed, protocols specified using finite state machines or Petri nets are not flexible in the sense that agents must respect the whole protocol from the beginning to the end. For this reason, we propose to specify these protocols by small computational dialogue games that can be considered as conversation policies that can be logically put together. Formally, a computational dialogue game is defined as follows:

Definition 4 (Computational Dialogue Game) *Let* $Action_{Ag1}$ *and* $Action_{Ag2}$ *be two communicative actions performed by* Ag_1 *and* Ag_2 *respectively, and let Cond be a formula from the logical language L. A computational dialogue game is a logical rule indicating that if* Ag_1 *performs* $Action_{Ag1}$*, and that Cond is satisfied, then* Ag_2 *will perform* $Action_{Ag2}$ *afterwards. This rule is expressed as follows:*

$$Action\_Ag_1 \xrightarrow{Cond} Action\_Ag_2$$

Cond is expressed in terms of the possibility of generating an argument from an agent's argumentation system.

3.3.5 Negotiation Dialogue Games

The negotiation protocol is specified as a set of computational dialogue games. In accordance with our commitment-based approach, the game moves are considered as actions that agents apply to commitments and to their contents (see Definition 1). Because we suppose that we have always two agents Ag_1 and Ag_2, a (SC) whose content is p will be denoted in the rest of this chapter $SC(p)$. The notation: $p \cap Arg\_Sys(Ag_1)$ is used to denote the fact that a propositional formula p can be generated from the argumentation system of Ag_1 denoted $Arg\_Sys(Ag_1)$. The formula $\neg(p \cap Arg\_Sys(Ag_1))$ indicates the fact that p cannot be generated from Ag_1's argumentation system. A propositional formula p can be generated from an agent's argumentation system, if this agent can find an argument supporting p. To simplify the formalism, the notation $Act'(Ag, SC(p))$ is used to indicate the action that agent Ag performs on the commitment $SC(p)$ or on its content ($Act' \in \{Create, Withdraw, Accept, Challenge, Refuse\}$). For the actions related to the argumentation relations, we write $Act\text{-}Arg(Ag, [SC(q)], SC(p))$.

This notation indicates that Ag defends (resp. attacks or justifies) the content of $SC(p)$ by the content of $SC(q)$ ($Act\text{-}Arg \in \{Defend, Attack, Justify\}$). In a general way, the notation $Act'(Ag, S)$ indicates the action that Ag performs on the set of commitments S or on the contents of these commitments, and the notation $Act\text{-}Arg(Ag, [S], SC(p))$ to indicate the argumentation-related action that Ag performs on the content of $SC(p)$ using the contents of S as support. The notation $Act\text{-}Arg(Ag, [S], S')$ is used to indicate that Ag performs an argumentation-related action on the contents of a set of commitments S' using the contents of S as supports.

Two types of dialogue games are distinguished: *entry game* and *chaining games*. The entry game allows the two agents to *open* the negotiation. The chaining games make it possible to combine dialogue games during the negotiation. The negotiation terminates when the exit conditions are satisfied (Fig. 5).

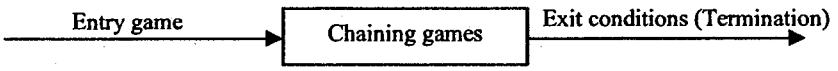
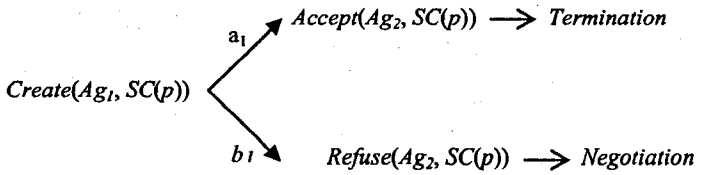


Figure 5 The General Form of the Protocol

A Entry Game

The *entry dialogue game* describing the entry conditions in our negotiation protocol about an offer represented by a propositional formula p is described as follows (*Specification 1*):



Where a_1 and b_1 are two conditions specified as follows:

$a_1 = p \cap Arg\_Sys\ (Ag_2)$

$b_1 = \neg p \cap Arg\_Sys(Ag_2)$

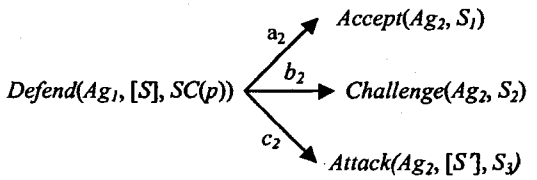
If Ag_2 has an argument for p, then it accepts the offer p (the content of SC (p)) and the conversation terminates as soon as it begins (Condition a_1). The negotiation starts when Ag_2 refuses the Ag_1's offer p because it has an argument against p (condition b_1).

B Defense Game

Once the two agents opened the negotiation, the initiator must defend its point of view in order to persuade the addressee. Consequently, a *defense game* should be played. Our protocol is

specified in such a way that the *negotiation dynamics* starts by playing a defense game. We call this type of negotiation *persuasive negotiation.*

This game is specified as follows (*Specification 2*):



Where:

$S=\{SC(p_i)/i=0,...,n\}$, p_i are propositional formulas.

$\bigcup_{i=1}^{3} S_i = S \bigcup SC(p)$, $S_i \bigcap S_j = \varnothing$, $i, j = 1,...,3 \,\&\, i \neq j$

By definition, *Defend* (Ag_1, [S], *SC* (p)) means that Ag_1 creates S in order to defend the content of *SC* (p). Formally:

Defend (Ag_1, [S], *SC* (p)) $\triangleq$ (*Create* (Ag_1, S) $\wedge$ S = *Create_Support* (Ag_1, *SC* (p)))

This definition is considered as an *assertional* description of the *Defend* action. Similar definitions for *Attack* and *Justify* actions can be proposed.

This specification indicates that according to the three conditions (a_2, b_2 and c_2); Ag_2 can accept a subset S_1 of S, challenge a subset S_2 and attack a third subset S_3. Sets S_i and S_j are mutually disjoint because Ag_2 cannot, for example, both accept and challenge the same commitment content. *Accept*, *Challenge* and *Attack* a set of commitment contents are defined as follows:

Accept (Ag_2, S_1) $\triangleq$ ($\forall i$, *SC* (p_i) $\in$ $S_1 \Rightarrow$ *Accept* (Ag_2, *SC* (p_i)))

Challenge (Ag_2, S_2) $\triangleq$ ($\forall i$, *SC* (p_i) $\in$ $S_2 \Rightarrow$ *Challenge* (Ag_2, *SC* (p_i)))

Attack (Ag_2, [S'], S_3) $\triangleq$ $\forall i$, *SC* (p_i) $\in$ $S_3 \Rightarrow \exists S'_j \subseteq S'$, *Attack* ($Ag_2$, [$S'_j$], *SC* ($p_i$))

Where: $S'_j = S'$. This indication means that any element of S' is used to attack one or more elements of S_3.

The conditions a_2, b_2 and c_2 are specified as follows:

$a_2 = \forall i, SC(p_i) \in S_1 \Rightarrow p_i \cap Arg\_Sys(Ag_2)$

$b_2 = \forall i, SC(p_i) \in S_2 \Rightarrow (\neg(p_i \cap Arg\_Sys(Ag_2)) \wedge \neg(\neg p_i \cap Arg\_Sys(Ag_2)))$

$c_2 = \forall i, SC(p_i) \in S_3 \Rightarrow \exists S'_j \subseteq S', Content(S'_j) = Support(Ag_2, \neg p_i)$

Where $Content(S'_j)$ indicates the set of contents of the commitments S'_j.

C Challenge Game

The *challenge game* is specified as follows (*Specification 3*):

$$Challenge(Ag_1, SC(p)) \xrightarrow{a_3} Justify(Ag_2, [S], SC(p))$$

Where the condition a_3 is specified as follows:

$a_3 = (Content(S) = Support(Ag_2, p))$

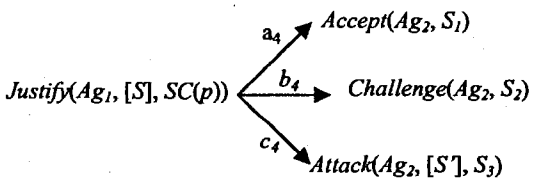
In this game, the condition a_3 is always true. The reason is that in accordance with the commitment semantics, an agent must always be able to defend the commitment it created [30].

D Justification Game

For this game we distinguish two cases:

Case1. $SC(p) \notin S$

In this case, Ag_1 justifies the content of its commitment $SC(p)$ by creating a set of commitments S. As for the Defend action, Ag_2 can accept, challenge and/or attack a subset of S. The specification of this game is as follows (Specification 4):



Where:

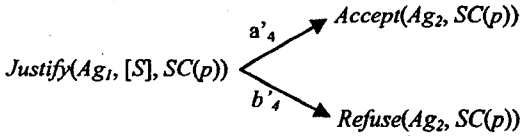
$S=\{SC(p_i)/i=0,...,n\}$, p_i are propositional formulas.

$\bigcup_{i=1}^{3} S_i = S \cup SC(p),\ S_i \cap S_j = \varnothing,\ i, j = 1,...,3 \& i \neq j$

$a_4 = a_2,\ b_4 = b_2,\ c_4 = c_2$

Case2. $\{SC(p)\} = S$

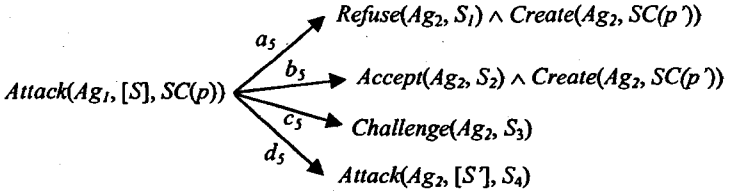
In this case, the justification game has the following specification (*Specification 5*):



Ag_1 justifies the content of its commitment *SC* (p) by itself (i.e. by p). This means that p is part of Ag_1's knowledge. Only two moves are possible for Ag_2: 1) *accept* the content of *SC* (p) if Ag_1 is a trustworthy agent for Ag_2 (a'_4), 2) if not, *refuse* this content (b'_4). Ag_2 cannot attack this content because it does not have an argument against p. The reason is that Ag_1 plays a justification game because Ag_2 played a challenge game.

E Attack Game

The *attack game* is specified as follows (*Specification 6*):



Where:

$S=\{SC(p_i)/i=0,...,n\}$, p_i are propositional formulas.

$\bigcup_{i=1}^{4} S_i = S \cup SC(p)$, $Card\ (S_i) = 1$, $S_i \cap S_j = \varnothing$, $i, j = 1,...,4 \& i \neq j$

The conditions a_5, b_5, c_5 and d_5 are specified as follows:

$a_5 = \exists i,\ SC\ (p_i) \in Create\_Support\ (Ag_2, SC\ (\neg q))$

Where $S_1 = \{SC\ (q)\}$

$b_5 = \forall i, SC(p_i) \in S_2 \Rightarrow p_i \cap Arg\_Sys(Ag_2)$

$c_5 = \forall i, SC(p_i) \in S_3 \Rightarrow (\neg(p_i \cap Arg\_Sys(Ag_2)) \wedge \neg(\neg p_i \cap Arg\_Sys(Ag_2)))$

$d_5 = \forall i, SC(p_i) \in S_4 \Rightarrow \exists S'_j \subseteq S'$,

$$Content(S'_j) = Support(Ag_2, \neg p_i) \wedge \nexists k, SC(p_k) \in Create\_Support(Ag_2, SC(\neg p_i))$$

Ag_2 refuses Ag_1's argument if Ag_2 already attacked this argument. In other words, Ag_2 refuses Ag_1's argument if Ag_2 cannot attack this argument since it *already* attacked it, and it cannot accept it or challenge it since it has an argument against this argument. We have only one element in S_1 since a refusal move is considered as an exit condition. The acceptance and the challenge actions of this game are the same as the acceptance and the challenge actions of the defence game. In the case of refusal and acceptance, Ag_2 can make a counteroffer p' by creating a new commitment. In this situation, Ag_1 will play an entry game by accepting or refusing the counteroffer. Finally, Ag_2 attacks Ag_1's argument if Ag_2 has an argument against Ag_1's argument, and if Ag_2 *did* not attack Ag_1's argument before. In d_5, the universal quantifier means that Ag_2 attacks all Ag_1's arguments for which it has an against-argument. The reason is that Ag_2 must act on all commitments created by Ag_1. The temporal aspect (the past) of a_5 and d_5 is implicitly integrated in *Create_Support* (Ag_2, $SC(\neg q)$) and *Create_Support* (Ag_2, $SC(\neg p_i)$).

F Termination

The protocol terminates either by a final acceptance or by a refusal. A final acceptance means that the two agents agree on a consensus. Because it is prohibited for the two agents to play the same move during the negotiation, the termination of the protocol is ensured.

3.4 Formal Analysis

3.4.1 Termination, Soundness, and Completeness

In this section, the formal properties of the negotiation protocol are discussed from a computational point of view. These properties are: termination (there is no deadlock in the protocol), soundness (the protocol specification is correct), and completeness (the protocol is complete with respect to the agents' knowledge bases).

Theorem 1 (Termination) For any set of dialogue games, the persuasive negotiation protocol always terminates.

Proof

The persuasive negotiation protocol is defined by the chaining of a finite set of dialogue games that can be played recurrently. Because the same move is prohibited during a conversation, and the content of communicative acts is finite in term of size, challenge and attack games are finite. In addition, because the agents' knowledge bases are finite and when an argument is justified by itself, the addressee could only accept or refuse (case 2 of justification game), then justification games are finite as well. Consequently, the protocol always converges toward executing either a final refusal or final acceptance.

Theorem 2 (Soundness) If the protocol terminates by a final acceptance or final refusal, then the outcome is in accordance with the union of the agents' knowledge bases.

Proof

According to the dialogue game specifications, if one of the participating agents plays the final acceptance move, this means that it has an argument supporting the addressee's argument advanced by playing a defence, attack, or justification game. Consequently, this agent has an argument supporting the last offer made by the addressee. Having this argument in the knowledge

base means that agreement is achieved. In the opposite case, if an agent plays a final refusal, then all the exchanged offers can not be supported by one of the two agents. This means that there is no argument from the two agents' knowledge bases supporting one of the offers. Consequently an agreement is not achieved.

The soundness property shows that the protocol is correct. However, what is important is to show that if an agreement is possible given the two agents' knowledge bases, then the protocol execution will achieve this agreement.

Theorem 3 (Completeness) If an agreement can be achieved from the agents' knowledge bases, then the protocol execution will result in achieving this agreement.

Proof

Let us suppose that from the union of the two agents' knowledge bases, it is possible to build an argument supporting a given offer p which is not attacked by another argument from the union. Consequently, this argument is accepted by the two agents. Let us show how this argument can be achieved when executing the protocol. We will use a proof by construction.

If p is the initial offer made, for example, by $Ag1$, then $Ag2$ will accept it in the entry game. So the agreement is achieved. If the initial offer is p' (p and p' are different but related because it is about the same topic) which is refused by $Ag1$, then $Ag2$ will defend it by proposing an argument. Ag1 will probably accept a part of this argument, challenge a second part, and attack a third part by possibly making a counter-offer. At this level, $Ag1$ can not completely accept the $Ag2$'s argument because its knowledge base is consistent. If this counter-offer is p (which is possible since $Ag1$'s argumentation system supports p), we are done, because it will be accepted by $Ag2$. If not, $Ag2$ will justify the challenge part and plays an attack game by possibly making a new counter-offer or refuse the attack and make a new counter-offer. If the counter-offer is p, then we are done. If not, $Ag1$ will play the same games. The process will continue until a counter-argument p is made by one of the two agents. There is a guarantee that p will be made, because if

not, one of the two agents will play a final refusal since according to Theorem 1 the protocol always terminates. This means that the final offer can not be supported by one of the two agents' knowledge bases and this agent can not make a counter-offer, which is contradictory with the initial hypothesis.

3.4.2 Complexity Analysis

It is proved that using first order logic and fully propositional logic for argumentative reasoning is not appropriate for automated negotiation since first order logic is semi-decidable and propositional logic is intractable (exponential time complexity) [44, 46]. Here it is proven that the protocol is efficient because the reasoning procedures are polynomial. This is because the logical language (Propositional Horn logic) is simpler and the dialogue games are simple logical rules. Because all these dialogue games are based on argumentation, and the decision parameters (the conditions associated to the rules) that agents use to combine these dialogue games are expressed in terms of the possibility of building arguments, the complexity of the protocol is determined by the complexity of building arguments. The following propositions present the different complexity results.

Proposition 1 Given a Horn knowledge base Γ, a subset $H \subseteq \Gamma$, and a formula h. checking whether (H, h) is a non-necessarily minimal argument is polynomial.

Proof

From the linear time algorithms for Horn satisfiability in [45], it follows that the Horn implication problem $H \vdash h$ is decidable in $O(|H| \times |h|)$ time. From the same result, it also follows that deciding whether H is consistent is polynomial.

Proposition 2 Given a Horn knowledge base Γ, and an argument (H,h) over Γ. Checking whether (H,h) is minimal is polynomial.

Proof

Let l be a literal. The following algorithm resolves the problem:

$\forall l \in H$, check if $H - \{l\} \vdash h$.. Because the implication problem is polynomial, we are done.

Proposition 3 Let Γ be a consistent Horn knowledge base, h a formula, and A the set of arguments over Γ

$\exists H \subseteq \Gamma : (H,h) \in A \Rightarrow \forall H' : H \subseteq H' \subseteq \Gamma, (H',h) \in A$

Proof

If (H,h) is an argument where H is a set of Horn formulas under the form c or $p_1 \vee p_2 \vee ... \vee p_n \rightarrow c$ where $p_1, p_2, ..., p_n$ are positive literals, then adding any Horn formula to H will result in a consistent set of formulas $H' : \Gamma \supseteq H' \supseteq H$. Since $H \vdash h$, it follows that $H' \vdash h$, hence the proposition.

Theorem 4 Given a consistent Horn knowledge base Γ and a formula h. Building an argument (H,h) from Γ is polynomial.

Proof

From Proposition 3, it follows that there is an argument supporting h iff $(\Gamma,h) \in A$. by Propositions 1 and 2, the theorem follows.

3.5 Implementation

In this section, a prototype implementation is described as proof of concept of the different dialogue games. The prototype is implemented using the Jack™ platform [43]. This language was chosen for three main reasons:

1- It is an agent-oriented language offering a framework for multi-agent system development. This framework can support different agent models.

2- It is built on top of and fully integrated with the Java programming language. It includes all components of Java and it offers specific extensions to implement agents' behaviors.

3- It supports logical variables and cursors. These features are particularly helpful when querying the state of an agent's beliefs. Their semantics is mid-way between logic programming languages with the addition of type checking Java style and embedded SQL.

Negotiating agents in the developed system are implemented as Jack™ agents, i.e. they inherit from the basic class Jack™ Agent. Their knowledge bases are implemented as Jack™ beliefsets. Beliefsets are used to maintain an agent's beliefs about the world. These beliefs are represented in propositional Horn logic and tuple-based relational model. The logical consistency of the beliefs contained in a beliefset is automatically maintained. The advantage of using beliefsets over normal Java data structures is that beliefsets have been specifically designed to work within the agent-oriented paradigm.

The agents' knowledge bases (KBs) contain two types of information: arguments and beliefs. Arguments have the form ([Support], Conclusion), where Support is a set of propositional Horn formulas and Conclusion is a propositional formula. Beliefs have the form ([Belief], Belief) i.e. Support and Conclusion are identical. The meaning of the propositional formulas (i.e. the ontology) is recorded in a beliefset whose access is shared between the two agents.

To open a dialogue game, an agent uses its argumentation system. The argumentation system allows this agent to seek in its knowledge base an argument for a given conclusion or for its

negation ("counter-argument"). For example, before creating a commitment SC(p), an agent must find an argument for p. This enables us to respect the commitment semantics by making sure that agents can always defend the content of their commitments.

Agent communication is done by sending and receiving messages. These messages are events that extend the basic Jack™ event: MessageEvent class. MessageEvents represent events that are used to communicate with other agents. Whenever an agent needs to send a message to another agent, this information is packaged and sent as a *MessageEvent*. A *MessageEvent* can be sent using the primitive: *Send*(*Destination*, *Message*). In our protocol, *Message* represents the action that an agent applies to a commitment or to its content, for example: *Create*(Ag_1, *SC* (p)), etc.

Our dialogue games are implemented as a set of events (*Message Events*) and plans. A plan describes a sequence of actions that an agent can perform when an event occurs. Whenever an event is posted and an agent chooses a task to handle it, the first thing the agent does is to try to find a plan to handle the event. Plans are methods describing what an agent should do when a given event occurs. Each dialogue game corresponds to an event and a plan. These games are not implemented within the agents' program, but as event classes and plan classes that are external to agents. Thus, each negotiating agent can instantiate these classes. An agent Ag_1 starts a dialogue game by generating an event and by sending it to the addressee Ag_2. Ag_2 executes the plan corresponding to the received event and answers by generating another event and by sending it to Ag_1. Consequently, the two agents can communicate by using the same protocol since they can instantiate the same classes representing the events and the plans. Figures 5 illustrate snapshots of the system.

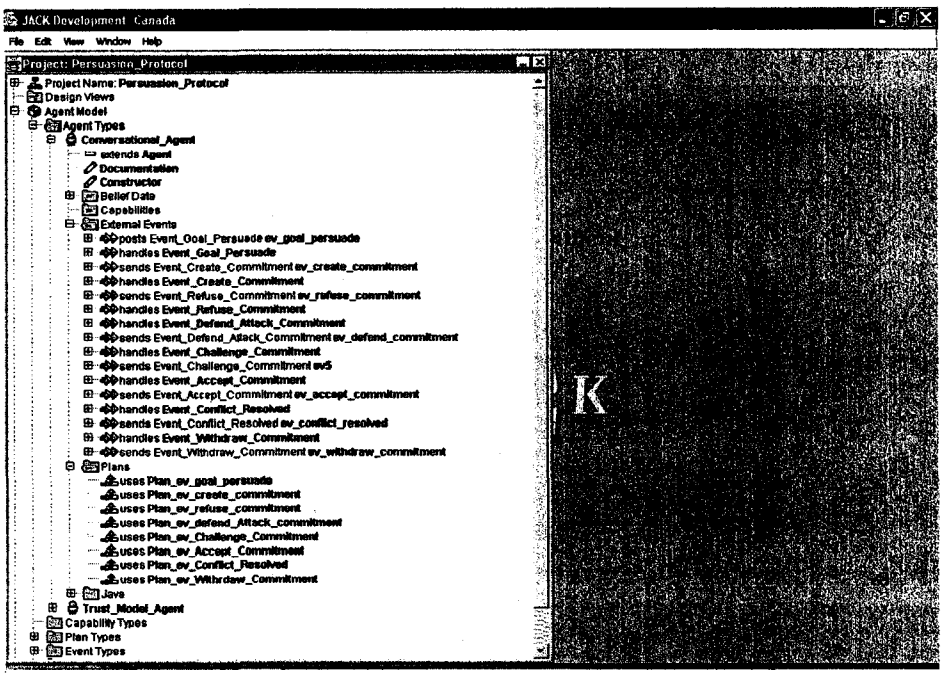


Figure 6. The System Data Structures

To start the entry game, an agent (initiator) chooses a goal that it tries to achieve. This goal is to persuade its interlocutor that a given propositional formula is true. For this reason, a particular event: BDI Event (Belief-Desire-Intention) is used. BDI events model goal-directed behaviour in agents, rather than plan-directed behaviour. What is important is the desired outcome, not the method chosen to achieve it. This type of events allows an agent to pursue long term goals.

3.6 Conclusion

Formal dialogue games are introduced as interaction games in which each agent plays a move in turn by performing utterances according to a pre-defined set of rules. In this chapter, we proposed an approach and a dialogue game protocol based upon persuasive argumentation for agent communication. In this approach, the agents' reasoning capabilities are linked to their ability to argue. The logical language used to specify the protocol has the advantage of being computationally efficient and expressing the public elements and the reasoning process that

allows agents to choose an action among several possible actions. Termination, soundness, and completeness properties of the proposed protocol are proven. Also, its computational complexity is discussed.

Chapter 4: A Trust Model based on Agent Categories

In this chapter, a new trust model addressing the security issue in multi-agent systems is introduced, by considering two factors. The first factor is about the classification of agents from a trust point of view using direct interactions. The second is about the categorization of the agents' chains through which the information is transmitted. Such a categorization is based upon the reliability of the agents in the chain.

4.1 Background

As stated in the introduction, agents in multi-agent systems need to communicate with one another in a seamless fashion just as in real societies. The research work presented in this thesis is based on a real society model used in 25% of the world today. Indeed, this work is inspired by a fundamental discipline in the Islamic faith known as "Ilm Al Hadith" or the knowledge of the authenticated saying of the prophet (Praise Be upon Him "P.B.U.H"). Ilm al Hadith is a form of investigation established by Muslim scholars in the 3rd century AH (9th century AD) to determine the validity of hadiths (sayings) of Muhammad's (P.B.U.H) statements, actions, and approbations as reported by various authorities.

There arose a need to accommodate a great diversity of cultures in the Muslim community, in the first two centuries of Islam, during the period of territorial expansion. The hadiths (information) multiplied in number and were often fabricated in order to create a normative past that could accommodate contemporary situations.

In the Islamic library, there exist six books that narrated the Hadiths (information) for authenticity purposes. Two of them are considered the most authenticated ones. The first is by Imam Bukhari (Sahih Bukhari) which is considered by far the most authenticated book after the Quran.

Imam Bukhari was among the first Islamic scholars that started writing down the prophet's (P.B.U.H) saying. He started his strict investigation on the narrators of such Hadiths (information / sayings of the prophet). And the second is by Imam Muslim (Sahih Muslim). Imam Muslim, the student of Imam Bukhari stated in the preface of his book (Sahih Muslim) that "*Narration from a reliable authority is mandatory in Islamic law and the science of Hadith in order to eliminate any doubt of perjury in narrating knowledge from the holy prophet*". He also stated in another chapter of the same book that "*declaration of the fact that the chain of authority is part of the deen (religion) and there should be no narration except from a reliable chain of authority*" [37].

Imam Muslim took great pains in collecting 300,000 traditions, and then after a thorough examination of them all retained only 4000; the genuineness of which is fully established. When we say that Imam Muslim collected 300,000 Hadiths and included only 4,000 in his compilation, it does not imply that he rejected the rest of the whole lot of the Prophet's sayings as being unreliable. Instead, what this means is that the words and deeds of the Holy Prophet (P.B.U.H) were transmitted to Imam Muslim through so many chains of transmission. Out of these many chains of transmission, he selected only 4,000 chains as being the most authentic and narrated the text on their authority [38].

Imam Ibn Sirin also states that "*the science of chain of authority and narration of Hadith is deen (religion) itself, you should check whom you are receiving your deen (religion) from* ".

Imam Muslim elaborated further from Imam Abdullah bin Mubarak, who says, "*between us and the people who receive from us are pillars of reliance and these are the chains of authority*" [37].

The Hadith is the text that contains the message of Islam and the teachings of the prophet. It is the substance of the Shariah (Islamic jurisdiction), whereas the chain consists of personalities. Reliance has been placed on the personalities over the actual content. Muslim

scholars attempted to determine forgeries or doubtful reports among the existing body of hadiths. They were bound however in principle to accept any textually reliable Hadith and so had to restrict themselves principally to the scrutiny of isnads—i.e., the chains of oral or written transmission by which the reliability of hadiths were determined.

The scholars therefore declared that all acceptable hadiths fall into four general categories: Category 1, Sahih (sound), those with a reliable and uninterrupted chain of transmission and a matn (Information) that does not contradict orthodox belief; Category 2, Hasan (good), those with an incomplete isnad or with transmitters of questionable authority; Category 3, Da'if (weak), those whose matn or transmitters are subject to serious criticism and Category 4, Mawdo (fabricated or forged).

Isnads (Chain of transmitters) are further evaluated according to the completeness of their chains: they may be unbroken and reliable all the way back to Muhammad (musnad) yet very short ('ali- high-), implying less likelihood of error; they may lack one authority in the chain of transmitters or may be missing two or more transmitters (mu'dal) or may have an obscure authority, referred to simply as "a man" (mubham).

The transmitters themselves, once established in the historical record as reliable men, determine further categories; the same tradition may have been handed down concurrently through several different isnads (mutawatir), indicating a long and sound history, or a Hadith may have been quoted by three different trustworthy authorities (mashhur) or by only one (ahad) [39].

This knowledge of authenticating Hadith is a meticulous and strict set of rules that deal with personalities, transmitters of Hadith. In multi-agent systems, the concern is how an agent can trust another. The idea is to use the reasoning behind this knowledge to introduce some concepts that serve the purpose of multi-agent interactions. The transmitted information in these interactions is composed of two factors, the information itself, and the chain of agents by which the information is obtained, similar to Ilm al Hadith.

4.2 Classification of Agents

In the proposed model, it is assumed that each agent in the environment has a network of agents that he trusts based on past interactions. Such a trust is relative to a given field that we denote by F. Each agent can then categorize the agents in his network according to the rate of successful interactions denoted by R. We define this categorization as a function Cat mapping two agents and a given field to a set of agent categories denoted by C. Let A be the set of agents, we define this function as follows:

$$Cat : A \times A \times F \rightarrow C$$

We distinguish four agent categories: *strong trustworthy agents* (*STA*), *trustworthy agents* (*TA*), *weak trustworthy agents* (*WTA*), and *untrustworthy agents* (*UA*), and unknown agent (*A?*).

Agents can use different policies to categorize other agents (in their network) using the rate R, which can be computed in different ways. For example, let $SI_{\alpha,\beta,f}$ be the number of successful interactions between two agents α and β about a subject f, and $TI_{\alpha,\beta,f}$ be the total number of their interactions about the same subject, this rate can be simply defined as follows:

$$R_{\alpha,\beta,f} = \frac{SI_{\alpha,\beta,f}}{TI_{\alpha,\beta,f}}$$

However, more complicated formulas can also be used, in order to give more importance to some interactions. For example, agents can evaluate their interactions according to a scale of n types numbered from 1 (the type of the most successful interactions) to n (the type of the less successful interactions), such that the first m interaction types ($m < n$) are successful. Examples of interaction types are: *very good*, *good*, *fair*, and *bad*. Let $I^i_{\alpha,\beta,f}$ be the number of interactions of type i between α and β about f, and w_i be the weight associated to this type. The rate R could be computed as follows:

$$R_{\alpha,\beta,f} = \frac{\sum_{i=1}^{m} w^i I^i_{\alpha,\beta,f}}{\sum_{i=1}^{n} w^i I^i_{\alpha,\beta,f}}$$

Using the rate R, an agent α can define the category of another agent β as follows:

$R_{\alpha,\beta,f} \geq v_1 \Rightarrow Cat(\alpha,\beta,f) = STA$

$v_2 \leq R_{\alpha,\beta,f} < v_1 \Rightarrow Cat(\alpha,\beta,f) = TA$

$v_3 \leq R_{\alpha,\beta,f} < v_2 \Rightarrow Cat(\alpha,\beta,f) = WTA$

$R_{\alpha,\beta,f} < v_3 \Rightarrow Cat(\alpha,\beta,f) = UA$

The values of the three variables $v_i (i = 1,...,3)$ depend on agents and the underlying application. For example, for some agents, these values could be instantiated as follows:

$v_1 = 0.9, v_2 = 0.8, v_3 = 0.5$ That is for a $v_1 = 0.9$, this implies that the percentage of successful interactions between α and β is at 90 %.

Agent categorizations influence the reliability of conveyed information. According to these categories, we distinguish four types of information: *sound*, *good*, *weak*, and *forged*. For example, the information transmitted by the first agent's type (*STA*) is expected to be sound (trusted information) and the one transmitted by the fourth type (*UA*) is expected to be forged (unaccepted information).

Let us consider the following predicates: $Convey(\beta,\alpha,i)$ indicates that the agent β conveys the information i to the agent α, and $f(i)$ indicates that the information i is related to the field f. Let $\mathcal{K}^\alpha$ be the set of knowledge whose agent α is sure about their truth. The following rules establish the relationship between the agent's type and reliability of conveyed information:

Rule 1: (Sound Information) is represented as follows:

$Convey(\beta,\alpha,i) \wedge f(i) \wedge Cat(\alpha,\beta,f) = STA \wedge \neg i \notin \mathcal{K}^\alpha$
$\Rightarrow Sound(i,\alpha)$

Rule 2: (Good Information) is represented as follows:

$$Convey(\beta,\alpha,i) \wedge f(i) \wedge Cat(\alpha,\beta,f) = TA \wedge \neg i \notin \mathcal{K}^{\alpha}$$
$$\Rightarrow Good(i,\alpha)$$

Rule 3: (Weak Information) is represented as follows:

$$Convey(\beta,\alpha,i) \wedge f(i) \wedge Cat(\alpha,\beta,f) = WTA \wedge \neg i \notin \mathcal{K}^{\alpha}$$
$$\Rightarrow Weak(i,\alpha)$$

Rule 4: (Forged Information) is represented as follows:

$$Convey(\beta,\alpha,i) \wedge f(i) \wedge Cat(\alpha,\beta,f) = UA \wedge \neg i \notin \ ^{\alpha}$$
$$\Rightarrow Forged(i,\alpha)$$

According to the first rule, if α considers β as an STA agent in the field f, β conveys the information i, which is related to the field f, and $\neg i$ is not a part of the α's sure knowledge, then the information i is considered sound by α. The other rules could be explained in the same way.

In the model, information can be conveyed not only by one agent but also through a set of agents under the form of chains. The idea is when an agent $^{\alpha}$ asks another agent β in his network about given information and if β does not have this information is his knowledge base ; agent β will then contact agents in his network in order to find an agent that will be able to convey the requested information. We have to take into consideration also that these agents could have *WTA*s and *UA*s in their network. Hence, when β sends a request for the information from his network he will send it to all the agents in his network. The conveyed information if found via his agents could be a result of a weak chain, since it might contain in it a *WTA* or a *UA*. Another situation could be that the agent is unknown, no one knows him, and they have not dealt with and as a result cannot rate him. Therefore, the chain will not be considered as conveyor of information. We will present these conditions when we talk about the quality and weight of the chain in question in later sections.

Let us first define the binary relation *Ask*.

Definition 1 (Ask relation): *Let* A *be the set of agents,* I *be the set of information expressed in a first order language,* α *and* β *be two agents in* A *and* i *be a formula representing some information in* $I \rightarrow \subseteq A \times A \times I$ *is a binary relation between agents about some information. This relation is defined as follows:* $\rightarrow \in \alpha, \beta, i$ *iff agent* α *asks agent* β *about information* i.

In the rest of the paper, we write $\alpha \rightarrow_i \beta$ *instead of* $\rightarrow \in \alpha, \beta, i$

Definition 2 (Chain of asking agents): *A chain* $C_{ask,i}^{\alpha_1,\alpha_2,...,\alpha_n}$ *of agents asking about information* i *is a finite sequence of agents* $\alpha_1, \alpha_2, ..., \alpha_n$ *such that:* $\forall j \ \ 1 \leq j \leq n : \alpha_j \rightarrow_i \alpha_{j+1}$.

We note that there is a temporal order in the chain of asking agents. To simplify the notation, this temporal aspect is not represented in the formulations, but we suppose its existence.

The predicate $Convey(\alpha_n, C_{ask,i}^{\alpha_1,\alpha_2,...,\alpha_n}, i)$. *indicates that agent* α_n *conveys information* i *to the agents forming the chain* $C_{ask,i}^{\alpha_1,\alpha_2,...,\alpha_n}$

Definition 3 (Chain of conveying agents): *A chain* $C_{con,i}^{\alpha_1,\alpha_2,...,\alpha_n}$ *of agents conveying information* i *is a chain* $C_{ask,i}^{\alpha_1,\alpha_2,...,\alpha_n}$ *such that:* $Convey(\alpha_n, C_{ask,i}^{\alpha_1,\alpha_2,...,\alpha_n}, i)$. α_1 *is called the source of the chain, and* α_n *its target.*

Definition 4 (Length of a Chain of conveying agents): *A chain of agents conveying information* i *is a finite sequence of agents* $\alpha_1, \alpha_2, ..., \alpha_n$ *such that:* $\forall j \ \ 1 \leq j \leq n : \alpha_j \rightarrow_i \alpha_{j+1}$ *and* $Convey(\alpha_n, \alpha_{n-1}, i)$. α_1 *is called the source of the chain, and* α_n *its target.*

Informally, the source of a chain of agents conveying information i is the agent that starts asking about this information, and the target is the final agent in the chain that has the information.

Because many chains could support contradictory information, we need to order chains according to their quality. To evaluate chain quality we should consider two factors: the weight of the chain and its length. The weight of a chain is defined in terms of the agent types in it. For

example a chain containing just *STA*s is better than a chain containing *STA*s and *TA*s, and a chain containing just *TA*s, is better than a chain containing some *WTA*s, even if it contains many *STA*s. Quality of chain = (weight of chain, **max-length of the chain)**

To define the weight of the chain we have to **define the weight of the agents in it**:

$$\forall \alpha_j \in C_{con,i}^{\alpha_1,\alpha_2,\ldots,\alpha_n}:$$
$$Weight(\alpha_j) = \min_{1 \le k < j}(Cat(\alpha_k, \alpha_j, f))$$

$$Weight(C_{con,i}^{\alpha_1,\alpha_2,\ldots,\alpha_n}) = \min_{1 \le j < n}(Weight(\alpha_j))$$

The weight of a chain is a pair **(bottleneck, max-length)**, for example, if a chain is formed of *STA*s and one *TA*, the weight of the chain will follow the weight of the weakest link and in our example it will follow the weight given to a *TA*.

4.3 Framework

The model suggests that if information is requested about a target agent (Ag_i), the agent will ask all the agents in his network to search for the information requested, as a result we should have different chains referring to the same information. The more chains the more reliable the information. Our model is represented in Figure 7.

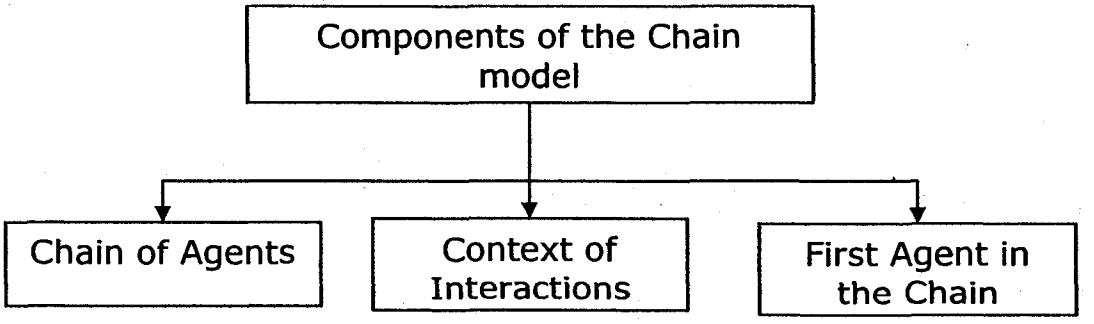


Figure 7 Chain Components

Chain of agents has an important and significant effect on the information transmitted; if the chain is long the information will lose its credibility, and the possibility of error is larger. The shorter the chain the better it is.

Let us assume that some information was transmitted from a chain formed by three agents and the same information was obtained from a chain of six agents; the information obtained from the three member chain will be chosen over the longer one.

Another importance is the fact that the chain carries some weight depending on the type of agents in it. The stronger the agent the better the chain and the reliability of the information is better as a result.

Context of the Interaction, it is important for the interaction purpose that the agent asked to supply information, make sure that these referee are familiar with the context of the interaction, otherwise, the information supplied by the agent will not gain credibility.

First agent in the chain is the asking agent. Its type will affect the chain, if the asking agent is *UA*, which implies that the chain will be composed of agents of his type, and hence the conveyed information is rejected. If on the other hand the asking agent is an *STA* or *TA* their agent network will contain more of their kind and again will influence the weight of the chain.

Based on the Proposed Chain model, it will be able to classify the information requested based on some conditions, such as the number of agents involved in the chain, the reliability of agents, etc... Figure 8 represents the classification of information.

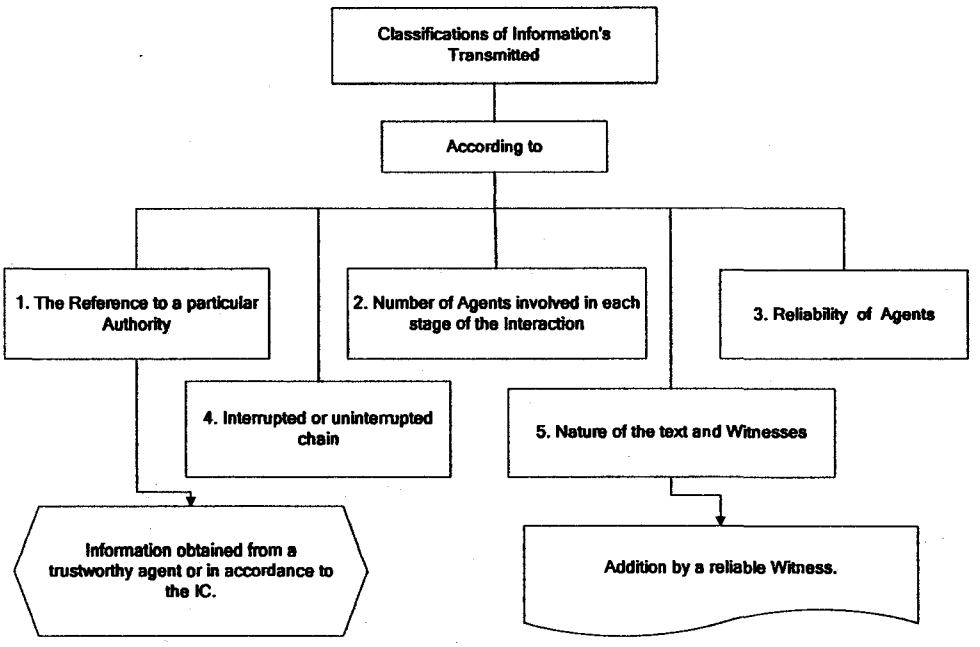


Figure 8 Classification of Information Transmitted

As can be seen in the above figure, the five point accordance will give rise to further classification of the information transmitted; the ones that were included in this thesis are *Sound, Good, Weak, and Forged.* In future work more classes will be proposed, which could be subclasses of the existing ones or simply new ones. The classification mentioned in the thesis (*STA, TA, WTA and UA*) will result in information that is (*Sound, Good, Weak or forged*) respectively. Those classes are under the reliability of agents. The other classes to be mentioned in the future will be generated from 1, 2, 4, and 5 in figure 8.

A successful interaction is strongly associated with agents. The agents involved in an interaction are introduced in a descending order of authentication or reliability:

- Strong trustworthy agent (*STA*).
- Trustworthy agent (*TA*).
- Weak trustworthy agent (*WTA*).
- Untrustworthy agent (*UA*).

Agents who have been unanimously described by the first two types contribute to sound and good information respectively. Agents described by the last type are likely to contribute in some fabricated or forged information and as a result an unacceptable source. The third type will contribute to weak information. The model is interested in both the information transmitted and the agents conveying information.

When for example an information is sound, we are not only talking about the information in itself, but also talking about the chain of agents that got us the information, a chain that is formed of only *STA*s that have dealt with each other and have significant amount of successful interaction (i.e. Ag_1 took an information from Ag_2, with whom he had a high rate of successful interactions, and in turns Ag_2 takes his information from Ag_3 with whom he had interacted successfully).

Property 1:

If two different chains having the same weight and conveying conflict information, the information is rejected unless a new chain support one or the other.

Example: Let us consider the two following chains having the same weight:

$Chain1: Ag_1 \rightarrow Ag_2 \rightarrow Ag_3 \rightarrow p$

$Chain2: Ag_{11} \rightarrow Ag_{22} \rightarrow Ag_{33} \rightarrow \neg p$

In this case, p and $\neg p$ are both rejected

Property 2:

When forming a chain, the classification of an agent (Ag_i) depends on the precedent agent (Ag_{i-1}). For example, Ag_i could be classified as a *TA* with respect to his precedent agent. However, he is classified as an *STA* with respect to (Ag_{i-n}). Hence the existence of other chains conveying

the same information and using some of the agents formed by the first chain but via a different route could be found and could carry a better weight.

There exist a lot of scenarios or propositions that the system encounters and solve. The model is to be used in any type of interactions especially in negotiation where trust plays an important role.

4.4 Conclusion

In this chapter, a new trust model for communicating agents has been introduced. It is a model based on a discipline found in the Islamic faith. The discipline is used to distinguish between good narrated sayings of the prophet and the bad or forged ones. The system uses several categorizations, one for the narrators themselves, another for the chain of narrators, another for the information obtained from these narrations. The latter discipline was the inspiration to start a categorizing system for interacting agents. The protocol distinguishes four categories of agents, and then with the use of direct interactions and the rate of successful ones, four types of interactions are introduced. Finally, these types of interactions influenced the types of conveying information.

Chapter 5: Implementation

5.1 Platform

Software agent technology is in high demand, many software companies are directing their attention on creating software that could be used for the creation of multi-agent environments. Some of these programs include Jack, Jade, and Jadex. The software we used in implementing the multi-agent environment is Jadex [41].

The Jadex system is based on the Belief Desire Intention (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 society, and artificial intelligence concepts.

The Jadex project [41] 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 concepts when programming plans commence, plans are plain Java classes, classes could include information such as sending message, or waiting for events. Jadex has included a special feature an intuitive OQL (Object Query Language are computer languages 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.

Michael Bratman, a philosophy professor was interested in what is known as the philosophy of action. This action theory is concerned with conjectures about the processes causing intentional human bodily movements of complex nature. This theory is the bases behind what is know in the agent world as BDI.

The agent world is the environment by which a set of agents exist and communicate. BDI stands for Beliefs, Desires, and intention. The Beliefs correspond to the knowledge or information that the agent has about the world. Desires represent the information that needs to be accomplished. Intentions are desires chosen for execution. In the execution phase, an agent searches for a plan that satisfies the intension. In order for the plans to be executed, it has to satisfy certain pre-conditions. These pre-conditions are checked against beliefs. The plans are steps that do the actual work of the agent, these steps may alter the beliefs which in turns might cause a change in the desires and will cause the desires to be unsatisfied; unsatisfied desires become intentions. Figure 9 represents BDI architecture.

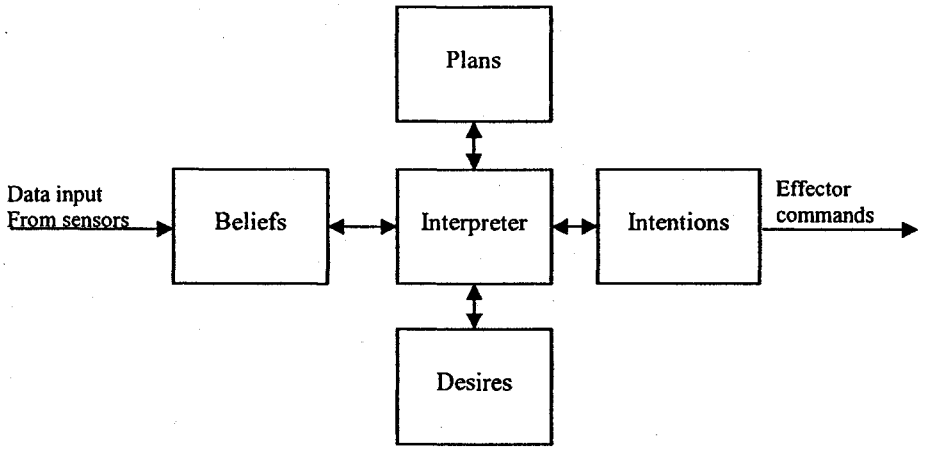


Figure 9 BDI Architecture

In order to develop an agent application in Jadex, one has two create two types of files; XML (eXtensible Markup Language) agent definition files (ADF) (see Appendix 1) and Java classes for the plan implementations. Plans describes the actions that an agent undertakes, the programmer 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.

5.2 Model Architecture

Our model is implemented on the Jadex software and as a result it follows the same agent architecture as the one presented in Figure 10. The figure shows how the execution on the agent level takes place in order to produce a message from the plans. Figure 10, will now be described specifically beliefs, goals, plans, and events.

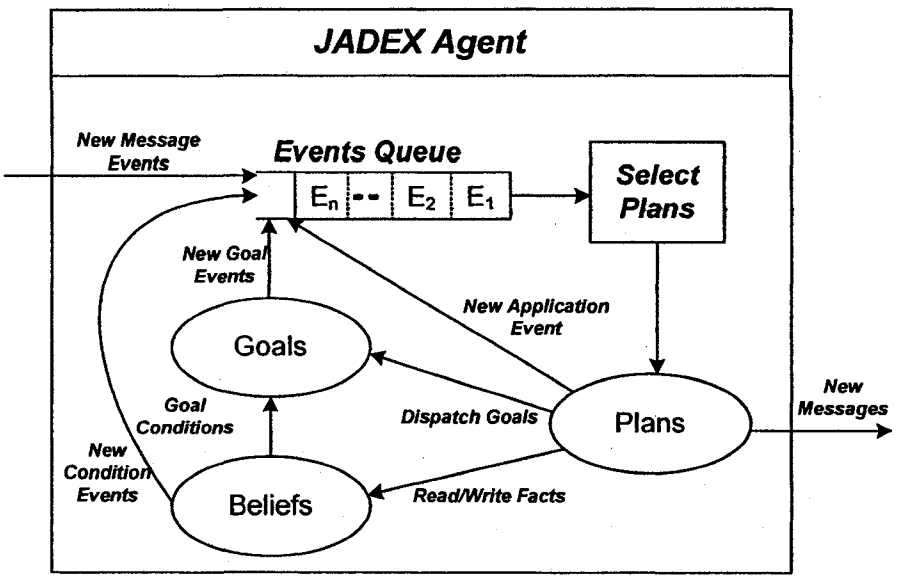


Figure 10 Agent Architecture

5.2.1 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. An example of an ADF (Agent Definition File) program is shown in Appendix 1, that program contains information for the beliefs, goals, and plans of an agent. The beliefbase contains strings that represent an identifier for a specific belief. Table 1 shows the summary of the proposed model beliefbase [41].

Belief Summary:	
Tuple	content_of_info Agent has the set of content information
AgentIdentifier	agent_list the set of all the agents in the community
AgentCategorization	agent_categorization the set of agent categorization (agent trust table)
ChainOfAgent	Chains the set of agent chain
Gui	Gui The Graphic interface of the agent.
Long	Time system current time

Table 1 Beliefbase Summary

5.2.2 Goals

Goals are a central concept in Jadex; they are concrete, momentary desires of an agent. Unlike traditional BDI systems, this 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 program will include the content of an agent goal (see Appendix 1) [41]. Table 2 shows the content of a goal that represents the model suggested.

Goals Summary:	
achievegoalref	ams_search_agents search the agents in AMS
performgoal	trust_table_init Initialize the agent categorization
performgoal	content_info_init Initialize the content information of the agent
achievegoal	search_content find the content in itself content table, if it is not in itself table, send message to looking for it; if it is in itself table, send message to all chains list tell the result.
achievegoal	send_message

Table 2 Goals Summary

5.2.3 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 agent developer needs to define. The head contains the conditions under which the plan may be executed and use is specified in the agent definition file (ADF) appendix 1. The body of the plan is a procedural recipe describing the actions to take in order to achieve a goal or react to some event, the body is written in JAVA. Table 3 shows the content of a plan that represents our model.

Plans Summary:	
Standard plan	InitTablePlan The plan initiates the trust table and content table.
Standard plan	FindContentPlan The plan finds if the content is in itself content table. If it is not in itself table, send message to looking for it; if it is in itself table, print the content, the value and the agentName. Trigger: messageevent"request_content
Standard plan	DealRequestPlan This plan will receive the message for asking some content information. If it is not in itself table, send message to looking for it; it is in itself table, send message to the first agent in the list tell the result. Trigger: messageevent "request_agent"
Standard plan	PrintChainPlan This plan will receive the message from the last agent in the chain and print the all agent name in the chain, the content value, the chain rate, which is the lowest rate in all the chain. Trigger: messageevent "inform_result"

Table 3 Plans Summary

5.2.4 Events

An important property of agents is the ability to react in a timely fashion to different kinds of events; again 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 [41]. Table 4 gives an event summary for our model implementation.

Events Summary:	
Message events	request_content direction = "receive"
Message events	request_agent direction="send-receive"
Message events	find_content direction = "receive"
Message events	inform_result direction = "send-receive"

Table 4 Events Summary

Figure 11 shows how the agent reason when the messages are received.

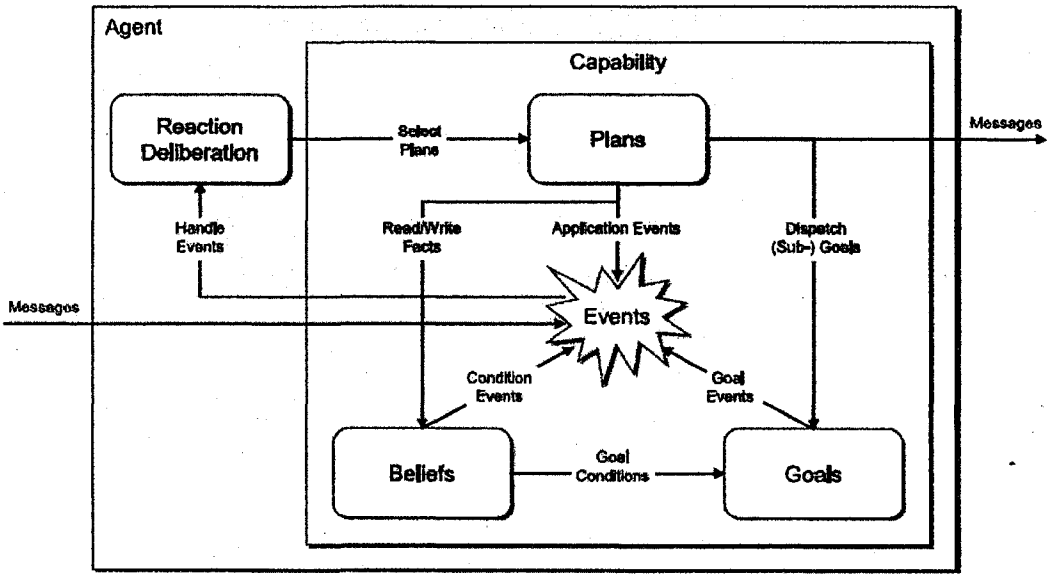


Figure 11 Reasoning Architecture

5.3 Agent Environment

For the implementation purposes, an environment with eight agents is used, where each agent has a network of agents that he considered reliable. Table 5 shows the characteristics of each agent in the environment; we will describe what the table represents.

	Agent Network	Rate	Category	Content of information
Ag_0	Ag_1	90	STA	R
	Ag_2	85	TA	
	Ag_5	75	TA	
Ag_1	Ag_2	95	STA	Q, S,
	Ag_6	85	TA	
	Ag_7	85	TA	
	Ag_8	45	UA	
Ag_2	Ag_1	90	STA	R, S, T
	Ag_4	35	UA	
	Ag_8	40	UA	
Ag_3	Ag_1	80	TA	Q, T
	Ag_6	90	STA	
	Ag_7	80	TA	
Ag_4	Ag_0	90	STA	T, S
	Ag_3	85	TA	
	Ag_2	75	TA	
Ag_5	Ag_1	90	STA	Q
	Ag_7	85	TA	
	Ag_8	30	UA	
Ag_6	Ag_2	90	STA	T, S, P
	Ag_3	85	TA	
	Ag_7	75	TA	
Ag_7	Ag_0	90	STA	T, S
	Ag_3	85	TA	
	Ag_2	45	UA	
Ag_8	Ag_0	90	STA	Q, P
	Ag_2	85	TA	
	Ag_7	45	UA	

Table 5 Agent Environment

$$R_{\alpha,\beta,f} \geq v_1 \Rightarrow Cat(\alpha,\beta,f) = STA$$

$$v_2 \leq R_{\alpha,\beta,f} < v_1 \Rightarrow Cat(\alpha,\beta,f) = TA$$

$$v_3 \leq R_{\alpha,\beta,f} < v_2 \Rightarrow Cat(\alpha,\beta,f) = WTA$$

$$R_{\alpha,\beta,f} < v_3 \Rightarrow Cat(\alpha,\beta,f) = UA$$

$v_1 = 0.9,\ v_2 = 0.8\ v_3 = 0.5.$

The categorization shown in table 5 follows the criteria given above. The content of the table tells us the following: if we take a look at the first row we distinguish the following information:

	Agent Network	Rate	Category	Content of information
Ag_0	Ag_1	90	STA	R, P, Q, S
	Ag_2	85	TA	
	Ag_5	75	TA	

This tells us that Ag_0 has three agents in his network, Ag_1, Ag_2, Ag_5, that he categorize them as STA, TA, and TA respectively according to the rate of successful interactions, and that Ag0 has in his beliefbase the information {R, P, Q, S}. The rest of the table can be explained the same way. In order to explain how the protocol works, let us assume that the staring agent is Ag_0, and the information requested is {T}. Let us also assume that the system will stop when the chain of agents reaches five agents, and that the agent will only ask the agents in his network that he considers as STA and TA, so that the information be considered authenticated and valid.

The information {T} is not in the content of Ag_0, which implies that Ag_0 will ask his STA and TA agents to see if they have it in their content. From now on we will present their communication with an arrow representation of the flow to get the information requested. Figure 12, shows the routes that the agents took to get the requested information.

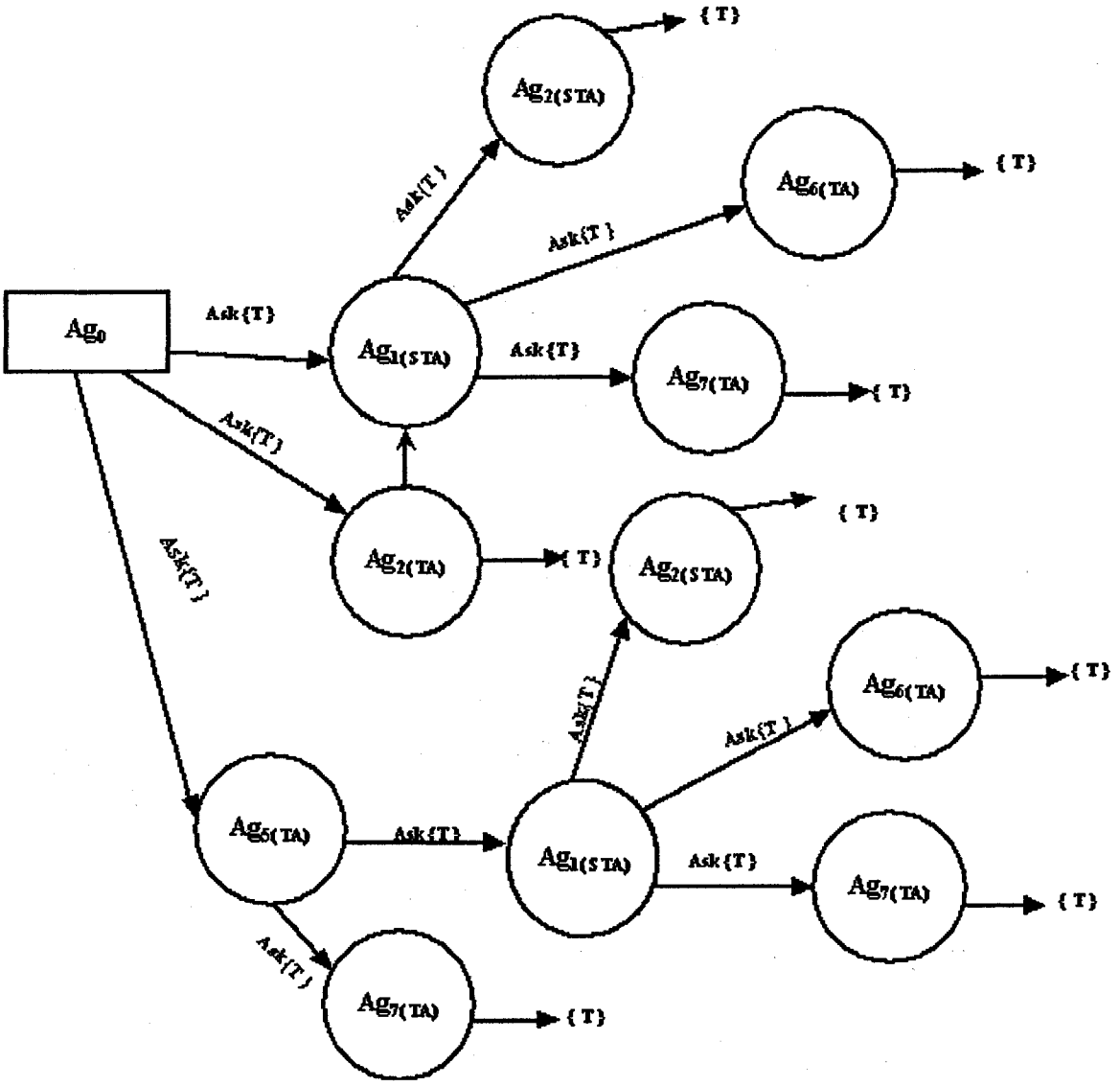


Figure 12 Agent Communication Route

From figure 12 we can deduce the following information:

Ten chains were found that could lead us to the information requested {T}:

1. Ag_0, Ag_1, Ag_2 this is a strong chain formed only by STA's

2. Ag_0, Ag_1, Ag_6 this is a good chain this it follows the bottleneck (TA)
3. Ag_0, Ag_1, Ag_7 this is a good chain
4. Ag_0, Ag_2 this is a good chain
5. Ag_0, Ag_2, Ag_1, Ag_6 good chain
6. Ag_0, Ag_2, Ag_1, Ag_7 good chain
7. Ag_0, Ag_5, Ag_7 good chain
8. Ag_0, Ag_5, Ag_1, Ag_2 good chain
9. Ag_0, Ag_5, Ag_1, Ag_6 good chain
10. Ag_0, Ag_5, Ag_1, Ag_7 good chain

These chains convey the information and are considered good chains. The strong chain (1) will be a good witness chain to back up the information of the other chains, and as a result, the information is authenticated and none of the chains are rejected.

The same procedure goes for any information that might be requested from the environment. When we implement this agent environment using Jadex, Jadex picks the starting agent randomly. In other word when the system is asked for the content {Q}, it will pick the starting agent randomly, and in one of the trial it picked agent 6 as a starting agent. Figure 12 shows the graphical representation of the route that Jadex will follow to get us the chains in this query.

ASKING Ag_6 FOR CONTENT {Q}

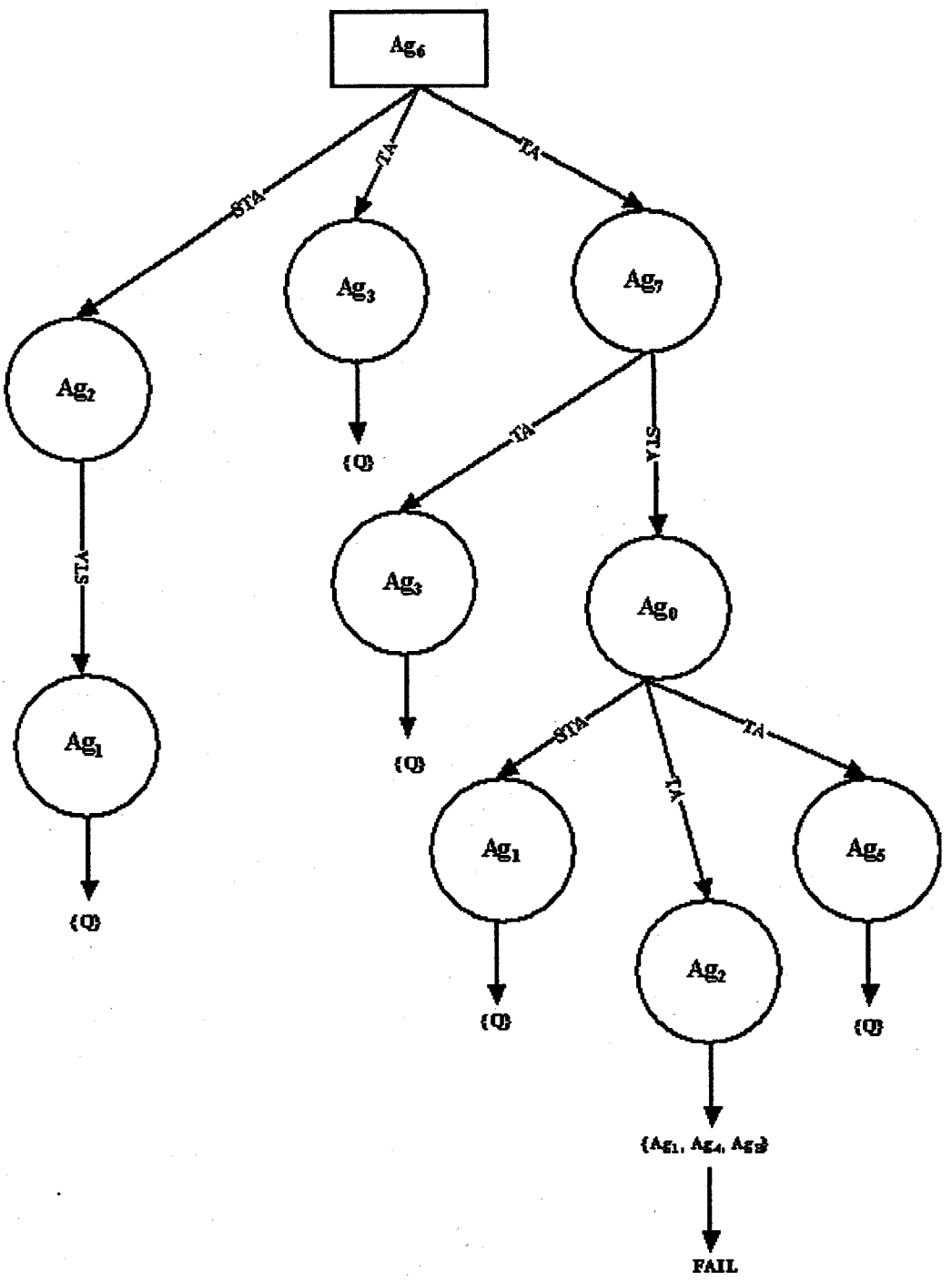


Figure 13 Asking *Ag*6 for {Q}

The implementation screenshot of figure 13 will be presented in appendix 3, Figure 15.

5.4 Algorithm

For the purpose of our implementation, and in order to avoid endless searches and loops, we decided to limit the chain of agents to four agents, after reaching four agents and the information is not obtained the searching process will end and the agent will look for another chain. When the agent needs to ask his trust agent belief, it will only ask the agents that are *STA* and *TA* as their ratings are high.

The following steps show how Jadex reason with the task on hand. How the program is able to find the chain of agents that supplies the requested information and in the next section an Agent Definition File (ADF) program of the implementation for the proposed model is shown:

Step 1:

The first step is to create an agent in Jadex. When the agent is asked for some content (information), the first thing that the agent will attempt is to search for the information his own belief bases (*content_of_info*), which are a tuple structure, *content* and *value*. This belief base stores all the content the agent believe in and the value of that content. If the information is found in his belief base, the process reaches termination. If the agent did not find the information we go to step 2.

Step 2:

When the agent fails to find the information in his belief base, the agent will send a message to its trust agents, who are found in the belief base named as "*agent_categorization*" and the type of trust is Strong Trustworthy (*STA*) and Trustworthy agents (*TA*), to require the information about this content. The message information includes the chain of agents, which is only the original agent right now, the agent chain length is 1 at this point, the content that it is looking for, and the value of this content, which is "false" as default because the original agent

does not know the value of this content, and the status, which is "open" (open means that the search is still on) at this time.

Step 3:

When the asked agent receives the message, it will first place itself in the chains last position and the chain length adds 1. Then it will follow the same procedure as in step 1, by comparing the content that other agents asked for from its own agent belief base.

If the content is in the agent belief base, the agent will send the final result message back to the first agent in the chain, which is the original agent. The message will include information (i.e. the chain of the agents, the chains length, the *content*, and *value*). The value of this content, which is true or false based on the value found in the agent belief base tuple, and the status, which will be "done" (done means the search is over).

If it is not in its belief base and the length of the chain is less than the maximum length (four), this agent will forward the message to all the agents in its trust agent beliefset (*agent_categorization*) belief base and the type of trust is *STA* and *TA*.

If it is not in its belief base and the length of the chain has arrived maximum length (four), then the agent will send final result message back to the original agent with status "fail" (fail means one of two things, either the information is not found or the chain of agents reached the maximum number).

Step 4:

The original agent gets the result messages and prints the results. In fact, the chain of agent may be more than one, up to four agents is allowed in our implementation.

Final step:

According to the messages that the original agent receives, the agent decides about the content and the value. Then agent will write the final result into its "*content_of_info*" belief base.

5.5 Results

From the screenshot (Appendix 2), we depicted three examples of the many results that the system was able to detect, the following explains the findings:

Jadex was able to find a chain of three agents with a rate of 0.9 and the result is true and done which signifies that the process is over. The query asked about content < Q > which was found in the beliefset.

```
the content (Q ) someone asked me is in my beliefset and the value is true
content is:the result is Ag2 Ag1 Ag0 3 Q 0.9 true done
```

Again the query was for a content Q, that does not belong in the beliefset, so it continue the search, however since for the purpose of the implementation we set a maximum for the agents in a chain to be four, the chain found contained five agents and the search failed although the rate is considered high at 0.84.

```
this content Q someone asked me is not in my beliefset.
content is:looking for chain Ag4 Ag3 Ag2 Ag1 Ag4 5 Q 0.84 false fail
```

The last of these results shows that the system was not able to find the content Q, it failed but did not stop the process, and a message of "open" means that the system is looking up with other trust agents, the process will end eventually by either finding the content or reaching the maximum number of agents.

```
I will ask my trust agent Ag3
content is:looking for chain Ag2 Ag1 2 Q 0.9 false open
```

5.6 Conclusion

The implementation of the model is carried out using a Java based system called Jadex. The Jadex system is based on the Belief Desire Intention (BDI) model and facilitates easy intelligent agent construction with sound software engineering foundations. To create agents, internal agent architecture and its environment should be considered as well as several artificial intelligence concepts. Since the model is based on direct interaction, when information is requested, it will be transmitted to the environment and Jadex will randomly pick a starting agent.

The starting agent will check in his beliefset whether the information requested is there or not. If not, the agent will ask his strong trustworthy agent and this will provide the chain of agents transmitting the information. For the implementation purposes and to limit the complexity, we have set the maximum chain to four agents, anything greater than the maximum is rejected and we have to look for other chains. In a normal setting, we will get more than three chains that will authenticate the requested information.

It is also assumed that the agent would only ask his STAs (Strong Trustworthy Agents) and TAs (Trustworthy Agents). However, the system allows the agent to ask WTAs (Weak Trustworthy Agents). For that purpose, some rules should be adopted to make the information accepted.

Chapter 6: Conclusion

6.1 Discussions

In this thesis, a new dialogue game approach and protocol for agent communication and a comprehensive trust model allowing agents to authenticate information conveyed by peers in the system have been proposed. The dialogue game protocol has the advantage of being computationally efficient and theoretically sound and complete. Unlike traditional agent communication protocols specified as finite state machines, the proposed protocol is flexible and compatible with agent autonomy.

In terms of trust, some trust models that are mainly used in open multi-agent systems, such as the FIRE, and Referral models have been introduced. In this chapter, we compare and present the pros and cons for each of these models and then compare them with our results.

The trust models presented in the second chapter lack the fulfillment to accomplish trust. They suggest a mathematical way assuming that agents are able to communicate and therefore they base their findings on assumptions. The referral model does not take into account that agent might lie. Instead, it assumes that agents will behave with honesty, which is not the case, because the existence of malicious agents defies the model. The FIRE model introduced the Certified Witness (References provided by other agents about the target agent's behaviours) similar to references needed when applying for a job. It could include reports from past interactions. The only advantage of certified witness is its availability. The setback of this is that the agent can choose which reference to use. In other words, an agent can ask the other agent to rate him according to the way he conducted the interactions with him. These ratings could be good while at the same time they might contain some bad ratings. Due to the fact that agents are autonomous and rational, they can choose whatever rating they want to serve their future interactions with other agents.

Any kind of interaction, whether it is between humans or software agents depends on two factors. The first factor is that negotiators need to trust each other, while the second factor is to find a language of communication between them. In the proposed model, we were able to combine both factors and to achieve the trust via agent categorization and chain of agents which resulted in the authentication of the information been transmitted.

The model suggests that any information (i) being transmitted depends on single or multiple agents, the agents need to be trusted or categorized in a way that the information would be accepted. Information may seem to be accurate and reasonable, but it needs an authentic chain with reliable agents to be acceptable. Sound information is the one which has a continuous chain, made up of trustworthy agents and which is found to be free from irregularities (i.e. in the text) or defects (i.e. in the chain). Good information is the information that does not contains disparaged agents in its chain, and which is transmitted through more than one chain. Weak information in most cases will not be considered unless it serves the interaction purposes and it is supported by witnesses. Forged information reflects the presence of untrustworthy agents (UAs) in the chain. This type of agents is to be avoided. Like in human societies, our model gives great importance to the chain of transmitters (agents).

The second factor, which is the communication language, is accomplished via the use of formal dialogue games. Once the trust is accomplished between different agents, dialogue games followed the rules set in the proposed trust model to carry on the negotiation.

The task of separating genuine information from apocryphal (of doubtful authenticity) information is as necessary as is that of removing weeds from a flower bed; as in the case of weeds, their identification and removal is not an easy task. Just as weeds cannot be left to flourish untouched, apocryphal information cannot be left in the system as they threaten the genuine information itself.

In this thesis, the proposed model will be able to pick up bad or untrustworthy agents from a pool of agents, much similar to finding weeds in a flower bed. By the use of agent

categorization we were able to prove whether the information is acceptable or not. We were also able to classify the chain of agents (similar to a list of referral agents) as being a strong or weak chain, which in turn resulted in accepting or rejecting the transmitted information.

From the implementation point of view, a prototype has been implemented as proof of concepts using agent-based programming. Jack and Jadex have been used to implement the dialogue games protocol and the trust model.

6.2 Future Work

As future work, we plan to investigate game theory and mechanism design to tackle the problem of agent lying. Combining trust and game theory is challenging from theoretical and practical point of view. The issue is how to use game theory notions such as equilibrium in trust settings. Using algorithmic mechanism design for trust purposes is also challenging particularly when there is no incentives in the environment. Another plan for future work is applying the proposed communication and trust model in real applications such as e-business and Web services. Computational complexity considerations in this case should be addressed.

Furthermore, to improve the agents' negotiation abilities in the proposed agent communication framework, agents can reason on the relevance of their offers and on the chance that their arguments can be accepted by the others. The idea is to go beyond the existing argumentation systems aiming simply to build an argument supporting a conclusion. The challenge is how to build a strong argument, ideally the stronger one. The idea we are investigating is to use a relevance-based reasoning in order to allow agents to optimize both their negotiation stances and the achievement of an agreement not only by justifying their choices, but by selecting the best choice that could be justified. Using rhetoric techniques combined with game theoretic and mechanism design strategies and some heuristics based on relevance theory seems a promising way. Agents can be equipped with "good" strategies enabling them to achieve their goals using an advanced reasoning on the utilities and the preferences of the other agents.

References

[1] R. Falcone, B.S. Firozabadi. The challenge of trust. Knowledge Engineering Review, vol. 14(1): 81-89, (1999).

[2] G. Weiss. Multi-Agent Systems: A Modern Approach to Distributed Artificial Intelligence, MIT Press, (1999).

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

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

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

[6] I. Rahwan, L.Sonenberg, N.R. Jennings, P. McBurney. STRATUM: A methodology for designing heuristic agent negotiation strategies. Applied Artificial Intelligence, vol. 21(10): 489-527, (2007).

[7] M. Luck, P. McBurney, and C. Preist. Agent Technology: Enabling Next Generation Computing. A Roadmap for Agent Based Computing. AgentLink II, pp. 94, (2003).

[8] D. N. Walton and E. C. W. Krabbe. Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning. SUNY Press, (1995).

[9] 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).

[10] P.McBurney and S.Parsons. Games the agents play: a formal framework for dialogues between autonomous agents. Journal of Logic, language and Information, vol. 11(3): 315-334, (2002).

[11] S. Parson, M. Wooldridge, and L. Amgoud. On the outcome of formal inter-agent dialogues. AAMAS Conference, pp. 616-623, (2003).

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

[13] E. Alonso. A formal framework for the representation of negotiation protocols. Inteligencia Artificial, vol 3: 30-49, (1997).

[14] L. Amgoud, S. Parson, and N. Maudet. Arguments, dialogue, and negotiation. In Proc. ECAI, pp. 338-342, (2000).

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

[16] D. Hitchcock. Some principles of rational mutual inquiry. In F. van Eemeren et al., editor, 2nd International Conference on Argumentation, pp. 236–243, (1991).

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

[18] M. Wooldridge. Semantic issues in the verification of agent communication languages. J. Autonomous Agents and Multi-Agent Systems, vol 3(1): 9-31, (2000).

[19] J. Forester. The Deliberative Practitioner: Encouraging Participatory Planning Processes. MIT Press, (1999).

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

[21] G. Brewka, Dynamic argument systems: A formal model of argumentation processes based on situation calculus. Journal of Logic and Computation, vol. 11(2): 257-282, (2001).

[22] M. Elvang-Goransson, J. Fox, and P. Krause, Dialectic reasoning with inconsistent information. In 9th Conference on Uncertainty in Artificial Intelligence, pp. 114-121, (1993).

[23] S. Kraus, K.P. Sycara, A. Evenchik, Reaching agreements through argumentation: a logical model and implementation. In Artificial Intelligence , vol. 104(1-2):1-69, (1998).

[24] L. Amgoud, L., Maudet, N., and Parsons, S. Modelling dialogues using argumentation. In Proc. of 4th Int. Conf. on Multi Agent Systems, pp. 31-38, (2000).

[25] J. Bentahar, B. Moulin, and B. Chaib-draa.. Commitment and argument network: a new formalism for agent communication, Agent Communication Languages, vol. 2922 of LNAI, pp. 146-165, (2004).

[26] 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).

[27] H. Prakken,. Relating protocols for dynamic dispute with logics for defeasible argumentation. In Synthese (127): 187-219, (2001).

[28] C. Castelfranchi,. Commitments: from individual intentions to groups and organizations. International Conference on Multi Agent Systems, pp. 41-48, (1995).

[29] N. Fornara, and M. Colombetti. Protocol specification using a commitment based ACL. In Advances in Agent Communication, LNAI 2922, pp 108-127, (2003).

[30] J. Bentahar, B.Moulin, J-J. Ch. Meyer, and B. Chaib-draa, A logical model for commitment and argument network for agent communication (extended abstract). In 3rd Int. J. Conf. on Autonomous Agents and Multi-Agent Systems AAMAS, pp. 792-799 (2004).

[31] U. Endriss, N. Maudet, F. Sadri, and F. Toni, Logic_based agent communication protocols. In Advances in Agent Communication, LNAI 2922, pp. 91-107, (2003).

[32] P.McBurney, and S. Parsons, Games that agents play: A formal framework for dialogues between autonomous agents. In Journal of Logic, Language, and Information, vol. 11(3): 1-22, (2002).

[33] Itu-t recommendation x.509. Information Technology. Open systems interconnection-the directory: Public-key and attribute certificate frameworks, (2000).

[34] C. Daly, IBM Corporation. A Trust Framework for the DoD Network-Centric Enterprise Services (NCES) Environment, (2004).

[35] T.D. Huynh, N.R. Jennings and R. Nigel, FIRE: An integrated trust and reputation model for open multi-agent systems, ECAI: pp.18-22, (2004).

[36] B. Yu and M. P. Singh. Searching social networks, International Joint Conference on Autonomous Agents and Multi-Agent Systems, pp 65-72 (2003).

[37] Imam Muslim in the Introduction to his Sahih . Sahih Muslim (ed. M.F. `Abdul Baqi, vol. 1:15, (1955).

[38] www.sunnah.org/history/Scholars/Imam_muslim

[39] 'Ilm Al-Hadith." Encyclopædia Britannica online,(2008).

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

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

[42] J. R. Searle. Speech Acts: an Essay in the Philosophy of Language. Cambridge University Press, (1969).

[43] The Agent Oriented Software Group. Jack 4.1. www.agent-software.com/ (2005).

[44] J. Bentahar, Z. Maamar, D. Benslimane, and Ph. Thiran,. "An argumentation framework for communities of web services". In IEEE Intelligent Systems, vol. 22(6): 75-83, (2007).

[45] W. Dowling, and J. H. Gallier, "Linear-time Algorithms for Testing the Satisfiability of Propositional Horn Theories". In Journal of Logic Programming, vol. 1(3): 267-284, (1984).

[46] S. Parsons, C. Sierra, N. Jennings, "Agents tat reason and negotiate by arguing". In Journal of Logic and Computation, vol. 8(3): 261-292, (1998).

Appendices

Appendix 1: Agent Definition File (ADF)

```
<!--
 <H3> Simulation for protocols for Agent Categorization and Trust in MAS. </H3>
-->
<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="Trust1" package="targetagent1">

        <imports>
                <import>jadex.adapter.fipa.SFipa</import>
                <import>jadex.runtime.*</import>
                <import>jadex.runtime.impl.*</import>
                <import>java.util.*</import>
                <import>jadex.planlib.*</import>
                <import>jadex.adapter.fipa.AgentIdentifier</import>
                <import>targetagent3.*</import>
          <import>jadex.util.*</import>
        </imports>

        <capabilities>
                <capability name="procap" file="jadex.planlib.Protocols"/>
                <capability name="dfcap" file="jadex.planlib.DF"/>
                <!--capability name="amscap" file="jadex.planlib.AMS"/-->
        </capabilities>

        <beliefs>
                <belief name="time" class="long" updaterate="1000">
                        <fact>System.currentTimeMillis()</fact>
                </belief>

          <!--belief name="max_chain_length" class="String">
            <fact> "4" </fact>
          </belief-->

                <beliefset name="content_of_info" class="Tuple">
               <fact>new Tuple("R","true")</fact>
               <fact>new Tuple("P","true")</fact>
                        <!--fact>InitTablePlan.getContentTable()</fact-->
                </beliefset>

                <belief name="agent_list" class="AgentIdentifier[]">
                </belief>
```

```
            <beliefset name="agent_categorization" class="AgentCategorization">
                <!--fact evaluationmode="dynamic"/-->
            </beliefset>

            <beliefset name="chains" class="ChainOfAgent">
                <!--fact evaluationmode="dynamic"/-->
            </beliefset>

            <beliefref name="cnp_filter" class="IFilter">
                <concrete ref="procap.cnp_filter"/>
            </beliefref>
        </beliefs>

        <goals>
            <achievegoal name="send_request" recur="true" recurdelay="10000">
            </achievegoal>

            <achievegoalref name="df_register">
                <concrete ref="dfcap.df_register"/>
            </achievegoalref>

            <achievegoalref name="de_deregister">
                <concrete ref="dfcap.df_deregister"/>
            </achievegoalref>

            <achievegoalref name="df_search">
                <concrete ref="dfcap.df_search"/>
            </achievegoalref>

            <performgoal name="trust_table_init" retry="true" exclude="never">
                <contextcondition>
                    $beliefbase.agent_categorization==null
                </contextcondition>
            </performgoal>

            <achievegoal name="find_content" recur="true" recurdelay="10000">
                <parameter name="chain" class="ChainOfAgent">
                </parameter>

        <targetcondition>ChainOfAgent.DONE.equals($goal.chain.getState())</targetcondition>
                <failurecondition>$beliefbase.time
>$goal.chain.getDeadline().getTime() </failurecondition>
            </achievegoal>
        </goals>

        <plans>
            <plan name="trust_table_plan">
                <body class="InitTablePlan"/>
                <trigger>
                    <goal ref="trust_table_init"/>
```

```
                </trigger>
            </plan>

            <plan name="find_content_plan">
                <body class="FindContentPlan"/>
                <trigger>
                    <messageevent ref="request_content"/>
                </trigger>
            </plan>

            <plan name="find_chain_plan">
                <body class="FindChaintPlan"/>
                <trigger>
                    <messageevent ref="request_agent"/>
                </trigger>
            </plan>
        </plans>

        <events>
            <messageevent name="request_content" type="fipa" direction="receive">
                <parameter name="performative" class="String" direction="fixed">
                    <value>SFipa.REQUEST</value>
                </parameter>
                <parameter name="content-start" class="String" direction="fixed">
                    <value>"looking for content"</value>
                </parameter>
                <parameter name="conversation-id" class="String">
                    <value>SFipa.createUniqueId($scope.getAgentName())</value>
                </parameter>
            </messageevent>

            <messageevent name="request_agent" type="fipa" direction="send_receive">
                <parameter name="performative" class="String">
                    <value> SFipa.INFORM</value>
                </parameter>
                <parameter name="conversation-id" class="String">
                    <value>
SFipa.createUniqueId($scope.getAgentName())</value>
                </parameter>
            </messageevent>

            <!--messageevent name="inform" type="fipa" direction="send_receive">
                <parameter name="performative" class="String">
                    <value> SFipa.INFORM</value>
                </parameter>
            </messageevent-->
        </events>

        <expressions>
            <expression name="search_agent_list"/>
```

```
        <expression name="query_content">
          select one $wordpair.get(1)
          from Tuple $wordpair in $beliefbase.content_of_info
          where $wordpair.get(0).equals($eword)
          <parameter name="$eword" class="String"/>

          <!--select one $wordpair.get(1) from Tuple $wordpaire in
$beliefbase.content_of_info
          where $wordpair.get(0).equals($content)
          <parameter name="$content" class="String"/-->
        </expression>

      </expressions>

      <properties>
            <property name="trust_service">

      SFipa.createAgentDescription(null,SFippa.createServiceDescription(null,"trust_service",
null))
            </property>
      </properties>

      <configurations>
            <configuration name="default">
                  <goals>
                        <initialgoal ref="df_register">
                              <parameter ref="description">

      <value>SFipa.createAgentDescription(null,SFipa.createServiceDescription("basic","trust
_service","Wei"))
                                          </value>
                                    </parameter>
                        </initialgoal>

                        <!--endgoal ref="de_deregister"/-->
                  </goals>
                  <plans>
                        <initialplan ref="trust_table_plan"/>
                  </plans>
            </configuration>
      </configurations>

</agent>
```

Appendix 2 More Results (Jadex Screenshot)

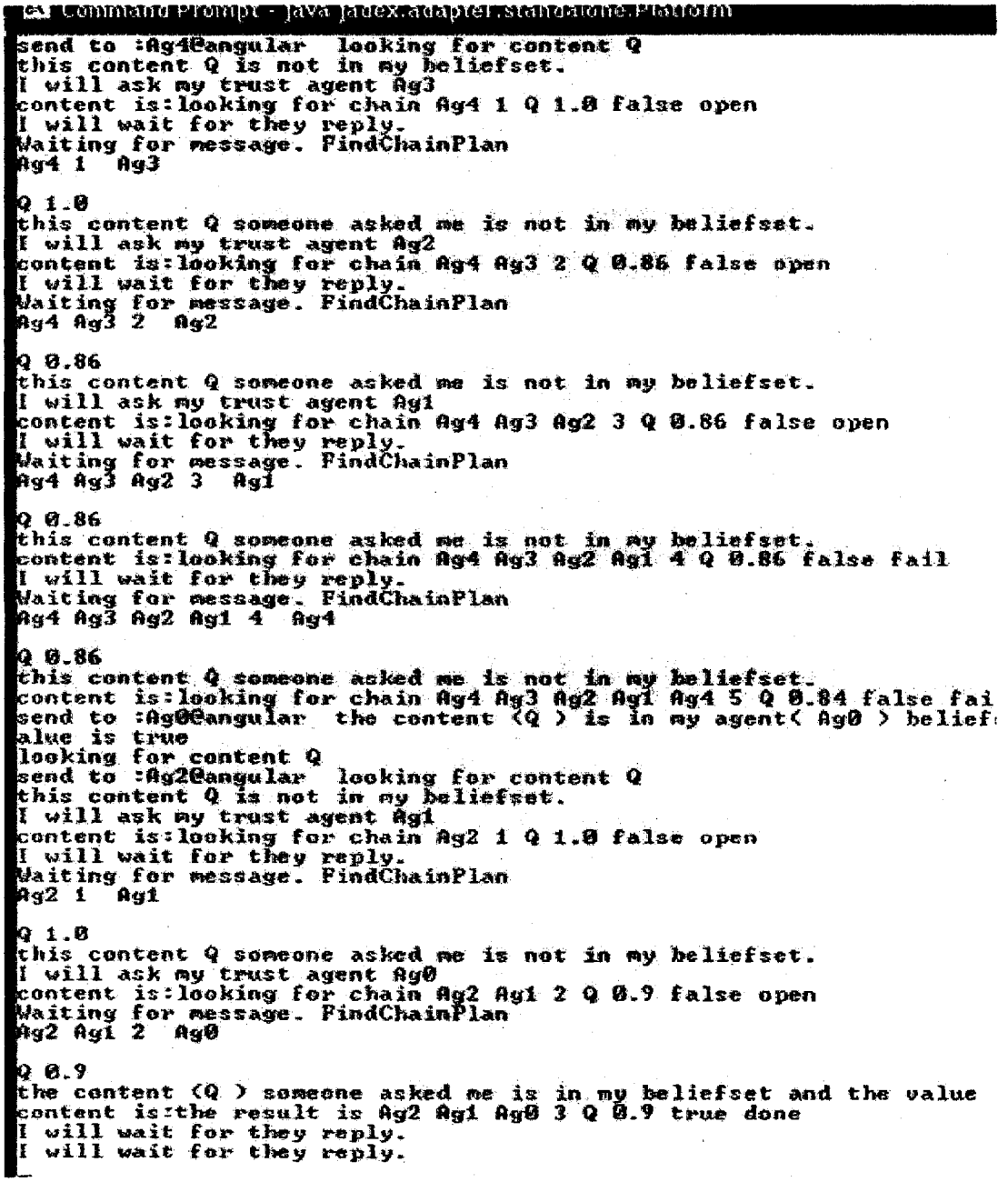


Figure 14 Screenshot of the Trust Implementation

Appendix 3 Asking Ag_6 for {Q} (Jadex Screenshot)

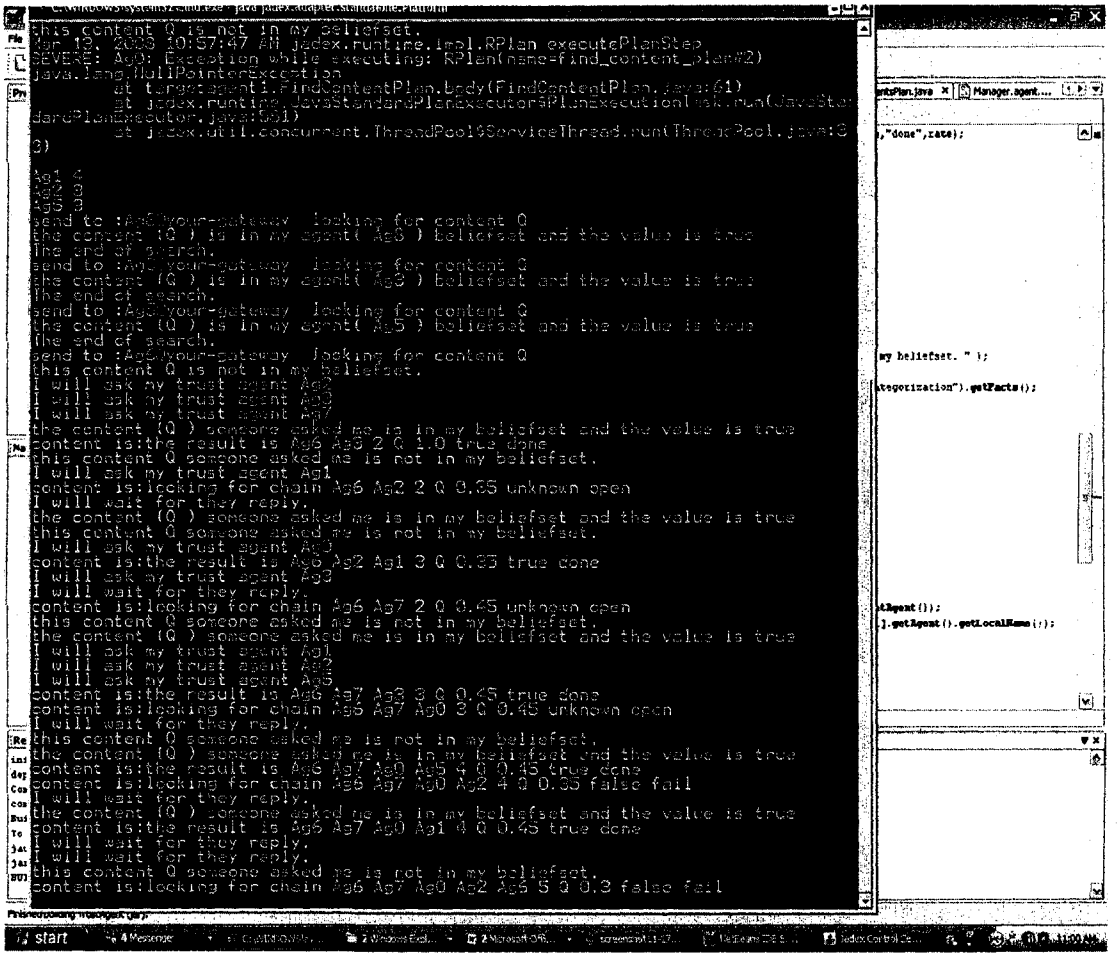


Figure 15 Jadex Screenshot

Glossary

A

ACT: A plan content language structured to be shared between independent plan generation and plan execution subsystems.

Actors (agent): Autonomous, interacting computing elements, which encapsulate a behaviour (data and procedures) and a process, and communicate by message-passing.

Agent: An autonomous, reactive, pro-active computer system, typically with a central locus of control, that is at least able to communicate with other agents via some kind of communication language.

Agent Architecture: A particular methodology for building agents. More generally, the term is used to denote a particular arrangement of data structures, algorithms, and control flows, which an agent uses in order to decide what to do. Agent architectures can be characterized by the nature of their decision making.

Agent Oriented Programming: An approach to building agents, which proposes programming them in terms of mentalist notions such as belief, desire, and intention.

Autonomy: Generally, autonomy means "under self-control."

B

BDI Architecture: A type of agent architecture containing explicit representations of beliefs, desires, and intentions. Beliefs are the information an agent has about its environment, which may be false; desires are those things that the agent would like to see achieved, and intentions are those things the agent is either committed to doing (intending to) or committed to bringing about (intending that).

C

Cognitive Concepts: Concepts applied in DAI that are inspired from folk psychology. These include the three BDI concepts, but also others such as know-how and commitments.

Coherence: The property or state of acting as a unit. A measure of how well a system behaves as a unit. Evaluation criteria for coherence are, e.g., efficiency, solution quality, and graceful degradation in the presence of failure.

Collaboration: Generally, "working together." Collaboration often refers to forms of high-level cooperation that require (the development of) a mutual understanding and a shared view of the task being solved by several interacting entities.

Commitments: Pledges by an agent to undertake a specified course of action.

Communication: How information is exchanged among agents but discount incidental interactions through the environment.

Content Language: The language in which the contents of message structures are encoded.

Coordination: Refers to the state of a community of agents in which actions of some agents fit in well with each other, as well as to the process of achieving this state.

D

Dialogue: Same as conversation.

Distributed Artificial Intelligence (DAI): Is the study and construction of systems composed of interacting, intelligent entities.

F

FIPA: Foundation for Intelligent Physical Agents; a consortium that is developing standards for agents.

G

Goals: A mutually consistent set of desires.

Group: A multi-agent system, especially one that is viewed (or acts or is intended to act) as a single agent.

I

Intentions: Goals that the agent is currently working on, i.e., those leading to the agent's actions.

Interaction: Generally, is everything that occurs "between" agents. (agent-agent interaction), and (agent-environment interaction).

K

KQML: Knowledge Query and Manipulation Language.

L

Learning (Distributed): Broadly speaking, learning refers to self-improvement of future behavior based on past experience. "Distributed" means that several entities (agents) are involved in the same learning process, where each entity contributes to the solution of the overall learning task according to its individual abilities or preferences.

Locution: The surface form of a speech act; that which is actually transmitted.

M

Modal Logic: The logic of necessity and possibility. This forms the basis of a number of the logics of BDI concepts.

Multi-Agent System: A system composed of multiple, interacting (see) agents. See also interaction.

N

Negotiation: Interaction among agents based on communication for the purpose of coming to an agreement. Negotiation has much to do with distributed conflict resolution and decision making.

O

Ontology: Generally, A specification of the objects, concepts, classes, functions and relationships in an area of interest.

P

P.B.U.H: Praise Be Upon Him

Predicate Logic: Propositional logic enhanced with variables and quantifiers to make statements about all or some objects in a given domain of discourse.

Protocol: A structured exchange of messages leading to some defined outcome. The rules of the interaction that describe what actions each agent can take at each time.

R

Rational: To behave in a way that is suitable or even optimal for goal attainment.

Reactive: (Of agent behaviour) Capable of maintaining an ongoing interaction with the environment, and responding in a timely fashion to changes that occur in it.

S

Social Ability: The ability to interact with other agents, typically by exchanging information via some language.

Software Agent: An agent that is implemented in software. See also interface agent. [GW]

Speech Act: A communication viewed as a combination of its locution.

Speech Act Theory: The view of natural language as actions. The basic claim is that utterances are an action that result in (or are intended by the speaker to result in) changes in the internal state a hearer.

T

Temporal Logic: Propositional logic augmented with operators to make claims about the truth of different conditions at different times. [2]