A New Quantitative Trust Model for Negotiating Agents using Argumentation

Jamal Bentahar[1], John-Jules Ch. Meyer[2]

[1] *Concordia University, Concordia Institute for Information Systems Engineering, Canada*
bentahar@ciise.concordia.ca

[2] *Utrecht University, Department of Information and Computing Sciences, The Netherlands*
jj@cs.uu.nl

Abstract

In this paper, we propose a new quantitative trust model for argumentation-based negotiating agents. The purpose of such a model is to provide a secure environment for agent negotiation within multi-agent systems. The problem of securing agent negotiation in a distributed setting is core to a number of applications, particularly the emerging semantic grid computing-based applications such as e-business. Current approaches to trust fail to adequately address the challenges for trust in these emerging applications. These approaches are either centralized on mechanisms such as digital certificates, and thus are particularly vulnerable to attacks, or are not suitable for argumentation-based negotiation in which agents use arguments to reason about trust.

Key words: Intelligent Agents, Negotiating Agents, Security, Trust.

1 Introduction

Research in agent communication protocols has received much attention during the last years. In multi-agent systems (MAS), protocols are means of achieving meaningful interactions between software autonomous agents. Agents use these protocols to guide their interactions with each other. Such protocols describe the allowed communicative acts that agents can perform when conversing and specify the rules governing a dialogue between these agents.

Protocols for multi-agent interaction need to be flexible because of the open and dynamic nature of MAS. Traditionally, these protocols are specified as finite state machines or Petri nets without taking into account the agents' autonomy. Therefore, they are not flexible enough to be used by agents expected to be autonomous in open MAS [16]. This is due to the fact that agents must respect the whole protocol specification from the beginning to the end without reasoning about them. To solve this problem, several researchers recently proposed protocols using dialogue games [6, 11, 15, 17]. Dialogue games are interactions between players, in which each player moves by performing utterances according to a pre-defined set of roles. The flexibility is achieved by combining different small games to construct complete and more complex protocols. This combination can be specified using logical rules about which agents can reason [11].

The idea of these logic-based dialogue game protocols is to enable agents to effectively and flexibly participate in various interactions with each other. One such type of interaction that is gaining increasing prominence in the agent community is negotiation. Negotiation is a form of interaction in which a group of agents, with conflicting interests, but a desire to cooperate, try to come to a mutually acceptable agreement on the division of scarce resources. A particularly challenging problem in this context is security. The problem of securing agent negotiation in a distributed setting is core to a number of applications, particularly the emerging semantic grid computing-based applications such as e-science (science that is enabled by the use of distributed computing resources by end-user scientists) and e-business [9, 10].

The objective of this paper is to address this challenging issue by proposing a new quantitative, probabilistic-based model to trust negotiating agents, which is efficient, in terms of computational complexity. The idea is that 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. Current approaches to trust fail to adequately address the challenges for trust in the emerging e-computing. These approaches are mostly centralized on mechanisms such as digital certificates, and thus are particularly vulnerable to attacks. This is because if some authorities who are trusted implicitly are compromised, then there is no other check in the system. By contrast, in the decentralized approach we propose in this paper and where the principals maintain trust in each other for more reasons than a single certificate, any "invaders" can cause limited harm before being detected. Recently, some decentralized trust models have been proposed [2, 3, 4, 7, 13, 19] (see [18] for a survey). However, these models are not suitable for argumentation-based negotiation, in which agents use their argumentation abilities as a reasoning mechanism. In addition, some of these models do not consider the case where false information is collected from other partners. This paper aims at overcoming these limits.

The rest of this paper is organized as follows. In Section 2, we present the negotiation framework. In Section 3, we present our trustworthiness model. We highlight its formulation, algorithmic description, and computational complexity. In Section 4, we describe and discuss implementation issues. In Sections 5, we compare our framework to related work, and in Section 6, we conclude.

2 Negotiation Framework

In this section, we briefly present the dialogue game-based framework for negotiating agents [11, 12]. These agents have a BDI architecture (Beliefs, Desires, and Intention) augmented with argumentation and logical and social reasoning. The architecture is composed of three models: the mental model, the social model, and the reasoning model. The mental model includes beliefs, desires, goals, etc. The social model captures social concepts such as conventions, roles, etc. Social commitments made by agents when negotiating are a significant component of this model because they reflect mental states. Thus, agents must use their reasoning capabilities to reason about their mental states before creating social commitments. The agent's reasoning capabilities are represented by the reasoning model using an argumentation system. Agents also have general knowledge, such as knowledge about the conversation subject. This architecture has the advantage of taking into account the three important aspects of agent communication: mental, social, and reasoning. It is motivated by the fact that conversation is a cognitive and social activity, which requires a mechanism making it possible to reason about mental states, about what other agents say (public aspects), and about the social aspects (conventions, standards, obligations, etc).

The main idea of our negotiation framework is that agents use their argumentation abilities in order to justify their negotiation stances, or influence other agent's negotiation stances considering interacting preferences and utilities. Argumentation can be abstractly defined as a dialectical process for the interaction of different arguments for and against some conclusion. Our negotiation dialogue games are based on formal dialectics in which arguments are used as a way of expressing decision-making [8, 14]. Generally, argumentation can 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 each other to make commitments. In contrast to traditional approaches to negotiation that are based on numerical values, argument-based negotiation is based on logic.

An argumentation system is simply a set of arguments and a binary relation representing the attack-relation between the arguments. The following definition, describe formally these notions. Here Γ indicates a possibly inconsistent knowledge base. $\vdash$ stands for classical inference and $\equiv$ for logical equivalence.

Definition 1 (Argument). *An argument is a pair* (H, h) *where h is a formula of a logical language 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.*

Definition 2 (Attack Relation). *Let* (H_1, h_1)*,* (H_2, h_2) *be two arguments.* (H_1, h_1) *attacks* (H_2, h_2) *iff* $h_1 \equiv \neg h_2$*.*

Negotiation dialogue games are specified using a set of logical rules. The allowed communicative acts are: Make-Offer, Make-Counter-Offer, Accept, Refuse, Challenge, Inform, Justify, and Attack. For example, according to a logical rule, before making an offer *h*, the speaker agent must use its argumentation system to build an argument (*H*, *h*). The idea is to be able to persuade the addressee agent about *h*, if he decides to refuse the offer. On the other side, the addressee agent must use his own argumentation system to select the answer he will give (Make-Counter-Offer, Accept, etc.).

3 Trustworthiness Model for Negotiating Agents

In recent years, several models of trust have been developed in the context of MAS [2, 3, 4, 13, 18, 19]. However, these models are not designed to trust argumentation-based negotiating agents. Their formulations do not take into account the elements we use in our negotiation approach (accepted and refused arguments, satisfied and violated commitments). In addition, these models have some limitations regarding the inaccuracy of the collected information from other agents. In this section we present our argumentation and probabilistic-based model to trust negotiating agents that overcome some limitations of these models.

3.1 Formulation

Let A be the set of agents. We define an agent's trustworthiness in a distributed setting as a probability function as follows:

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

This function associates to each agent a probability measure representing its trustworthiness in the domain *D* according to another agent. To simplify the notation, we omit the domain *D* from the *TRUST* function because we suppose that is always known. Let *X* be a random variable representing an agent's trustworthiness. To evaluate the trustworthiness of an agent Ag_b, an agent Ag_a uses the history of its interactions with Ag_b. Equation 1 indicates how to calculate this trustworthiness as a probability measure (*number of successful outcomes* / *total number of possible outcomes*).

$$TRUST(Ag_b)_{Ag_a} = \frac{Nb\_Arg(Ag_b)_{Ag_a} + Nb\_C(Ag_b)_{Ag_a}}{T\_Nb\_Arg(Ag_b)_{Ag_a} + T\_Nb\_C(Ag_b)_{Ag_a}} \quad \textbf{(1)}$$

$TRUST(Ag_b)_{Ag_a}$ indicates the trustworthiness of Ag_b according to Ag_a's point of view.
$Nb\_Arg(Ag_b)_{Ag_a}$ is the number of Ag_bs' arguments that are accepted by Ag_a.
$Nb\_C(Ag_b)_{Ag_a}$ is the number of satisfied commitments made by Ag_b towards Ag_a.
$T\_Nb\_Arg(Ag_b)_{Ag_a}$ is the total number of Ag_bs' arguments towards Ag_a.
$T\_Nb\_C(Ag_b)_{Ag_a}$ is the total number of commitments made by Ag_b towards Ag_a.

All these commitments and arguments are related to the domain *D*. The basic idea is that the trust degree of an agent can be induced according to how much information acquired from him has been accepted as belief in the past. Using the number of accepted arguments when computing the trust value reflects the agent's knowledge level in the domain *D*. Particularly, in the argumentation-based negotiation, the accepted arguments capture the agent's reputation level. If some argument conflicts in the domain *D* exist between the two agents, this will affect the confidence they have about each other. However, this is related only to the domain *D*, and not generalized to other domains in which the two agents can trust each other. In a negotiation setting, the existence of argument conflicts reflects a disagreement in the perception of the negotiation domain. Because all the factors of Equation 1 are related to the past, this information number is finite.

Trustworthiness is a dynamic characteristic that changes according to the interactions taking place between Ag_a and Ag_b. This supposes that Ag_a knows Ag_b. If not, or if the number of interactions is not sufficient to determine this trustworthiness, the consultation of other agents becomes necessary.

As proposed in [1, 2, 3], each agent has two kinds of beliefs when evaluating the trustworthiness of another agent: local beliefs and total beliefs. Local beliefs are based on the direct interactions between agents. Total beliefs are based on the combination of the different testimonies of other agents that we call *witnesses*. In our model, local beliefs are given by Equation 1. Total beliefs require studying how different probability measures offered by witnesses can be combined. We deal with this aspect in the following section.

3.2 Estimating Agent's Trustworthiness

Let us suppose that an agent Ag_a wants to evaluate the trustworthiness of an agent Ag_b with who he never (or not enough) interacted before. This agent must ask agents he knows to be trustworthy (we call these agents *confidence agents*). To determine whether an agent is confident or not, a trustworthiness threshold w must be fixed. Thus, Ag_b will be considered trustworthy by Ag_a iff $TRUST(Ag_b)_{Ag_a}$ is higher or equal to w. Ag_a attributes a trustworthiness measure to each confidence agent Ag_i. When he is consulted by Ag_a, each confidence agent Ag_i provides a trustworthiness value for Ag_b if Ag_i knows Ag_b. Confidence agents use their local beliefs to assess this value (Equation 1). Thus, the problem consists in evaluating Ag_b's trustworthiness using the trustworthiness values transmitted by confidence agents. Fig. 1 illustrates this issue.

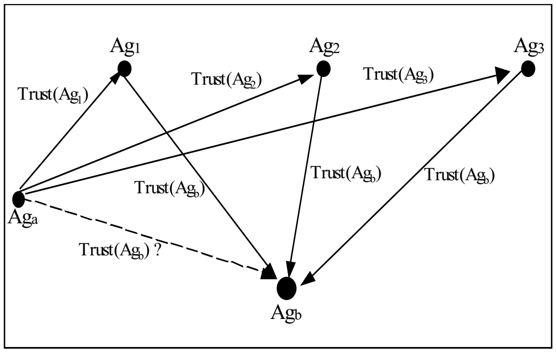


Fig. 1. Problem of measuring Ag_b's trustworthiness by Ag_a

We notice that this problem cannot be formulated as a problem of conditional probability. Consequently, it is not possible to use *Bayes' theorem* or *total probability theorem*. The reason is that events in our problem are not mutually exclusive, whereas this condition is necessary for these two theorems. Here an event is the fact that a confidence agent is trustworthy. Consequently, events are not mutually exclusive because the probability that two confidence agents are at the same time trustworthy is not equal to 0.

To solve this problem, we must investigate the distribution of the random variable *X* representing the trustworthiness of Ag_b. Since *X* takes only two values: 0 (the agent is not trustworthy) or 1 (the agent is trustworthy), variable *X* follows a Bernoulli distribution $\beta(1, p)$. According to this distribution, we have Equation 2:

$$E(X) = p \qquad \textbf{(2)}$$

where *E*(*X*) is the expectation of the random variable *X and p* is the probability that the agent is trustworthy. Thus, *p* is the probability that we seek. Therefore, it is enough to evaluate the expectation *E*(*X*) to find $TRUST(Ag_b)_{Ag_a}$. However, this expectation is a theoretical mean that we must estimate. To this end, we can use the *Central Limit Theorem* (CLT) and the *law of large numbers*. The CLT states that whenever a random sample of size *n* ($X_1, \ldots X_n$) is taken from any distribution with mean μ, then the sample mean (X_1 + ... +X_n)/*n* will be approximately normally distributed with mean μ. As an application of this theorem, the arithmetic mean (average) (X_1+...+ X_n)/*n* approaches a normal distribution of mean μ, the expectation and standard deviation $\sigma/\sqrt{n}$ .Generally, and according to the law of large numbers, the expectation can be estimated by the weighted arithmetic mean.

Our random variable *X* is the weighted average of *n* independent random variables X_i that correspond to Ag_b's trustworthiness according to the point of view of confidence agents Ag_i. These random variables follow the same distribution: the Bernoulli distribution. They are also independent because the probability that Ag_b is trustworthy according to an agent Ag_t is independent of the probability that this agent (Ag_b) is trustworthy according to another agent Ag_r. Consequently, the random variable *X* follows a normal distribution whose average is the weighted average of the expectations of the independent random variables X_i. The mathematical estimation of expectation *E*(*X*) is given by Equation 3.

$$M_0 = \frac{\sum_{i=1}^{n} TRUST(Ag_i)_{Ag_a} TRUST(Ag_b)_{Ag_i}}{\sum_{i=1}^{n} TRUST(Ag_i)_{Ag_a}} \qquad \textbf{(3)}$$

The value M_0 represents an estimation of $TRUST(Ag_b)_{Ag_a}$. Equation 3 does not take into account the number of interactions between confidence agents and Ag_b. This number is an important factor because it makes it possible to promote information coming from agents knowing more Ag_b. In addition, an other factor might be used to reflect the timely relevance of transmitted information. This is because the agent's environment is dynamic and may change quickly. The idea is to promote recent information and to deal with out-of-date information with less emphasis. Equation 4 gives us an estimation of $TRUST(Ag_b)_{Ag_a}$ if we take into account these factors and we suppose that all confidence agents have the same trustworthiness.

$$M_1 = \frac{\sum_{i=1}^{n} N(Ag_i)_{Ag_b} TR(Ag_i)_{Ag_b} TRUST(Ag_b)_{Ag_i}}{\sum_{i=1}^{n} N(Ag_i)_{Agb} TR(Ag_i)_{Ag_b}} \qquad \textbf{(4)}$$

The factor $N(Ag_i)_{Ag_b}$ indicates the number of interactions between a confidence agent Ag_i and Ag_b. This number can be identified by the total number of Ag_b's commitments and arguments. The factor $TR(Ag_i)_{Ag_b}$ represents the timely relevance coefficient of the information transmitted by Ag_i about Ag_b 's trust (*TR* denotes *Timely Relevance*). We denote here that removing $TR(Ag_i)_{Ag_b}$ from the Equation 4, results in the classical probability equation used to calculate the expectation *E*(*X*).

In our model, we assess the factor $TR(Ag_i)_{Ag_b}$ by using the function defined in Equation 5. We call this function: the *Timely Relevance function*.

$$TR(\Delta t_{Ag_i}^{Ag_b}) = e^{-\lambda \ln(\Delta t_{Ag_i}^{Agb})} \qquad \textbf{(5)}$$

Δt is the time difference between the current time and the time at which Ag_i updates its information about Ag_b's trust. λ is an application-dependant coefficient. The intuition behind this formula is to use a function decreasing with the time difference (Fig. 2). Consequently, the more recent the information is, the higher is the timely relevance coefficient. The function ln is used for computational reasons when dealing with large numbers. Intuitively, the function used in Equation 5 reflects the reliability of the transmitted information. Indeed, this function is similar to the well known *reliability function* for systems engineering ($R(t) = e^{-\lambda t}$).

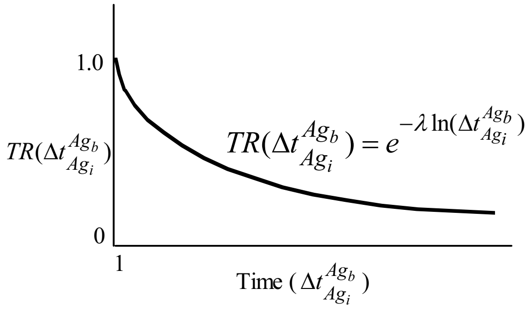


Fig. 2. The timely relevance function

The combination of Equation 3 and Equation 4 gives us a good estimation of $TRUST(Ag_b)_{Ag_a}$ (Equation 6) that takes into account the four most important factors: (1) the trustworthiness of confidence agents according to the point of view of Ag_a; (2) the Ag_b's trustworthiness according to the point of view of confidence agents; (3) the number of interactions between confidence agents and Ag_b; and (4) the timely relevance of information transmitted by confidence agents. This number is an important factor because it makes it possible to highlight information coming from agents knowing more Ag_b.

$$M_2 = \frac{\sum_{i=1}^{n} TRUST(Ag_i)_{Ag_a} N(Ag_i)_{Ag_b} TR(Ag_i)_{Ag_b} TRUST(Ag_b)_{Ag_i}}{\sum_{i=1}^{n} TRUST(Ag_i)_{Ag_a} N(Ag_i)_{Ag_b} TR(Ag_i)_{Ag_b}} \qquad \textbf{(6)}$$

The way of combining Equation 3 (M_0) and Equation 4 (M_1) in the calculation of Equation 6 (M2) is justified by the fact that it reflects the mathematical expectation of the random variable X representing the Ag_b's trustworthiness. This equation represents *the sum of the probability of each possible outcome multiplied by its payoff.*

This Equation shows how trust can be obtained by merging the trustworthiness values transmitted by some mediators. This merging method takes into account the proportional relevance of each trustworthiness value, rather than treating them equally.

According to Equation 6, we have:

$$\forall i, TRUST(Ag_b)_{Ag_i} < w$$
$$\Rightarrow M < w.\frac{\sum_{i=1}^{n} TRUST(Ag_i)_{Ag_a} N(Ag_i)_{Ag_b} TR(Ag_i)_{Ag_b}}{\sum_{i=1}^{n} TRUST(Ag_i)_{Ag_a} N(Ag_i)_{Ag_b} TR(Ag_i)_{Ag_b}}$$
$$\Rightarrow M < w$$

Consequently, if all the trust values sent by the consulted agents about Ag_b are less than the threshold w, then Ag_b can not be considered as trustworthy. Thus, the well-known *Kyburg's lottery paradox* can never happen. The lottery paradox was designed to demonstrate that three attractive principles governing rational acceptance lead to contradiction, namely that:
1. it is rational to accept a proposition that is very likely true;
2. it is not rational to accept a proposition that you are aware is inconsistent; and
3. if it is rational to accept a proposition A and it is rational to accept another proposition B, then it is rational to accept $A \wedge B$,
are jointly inconsistent. In our situation, we do not have such a contradiction.

To assess M, we need the trustworthiness of other agents. To deal with this issue, we propose the notion of *trust graph*.

3.3 Trust Graph

In the previous section, we provided a solution to the trustworthiness combination problem to evaluate the trustworthiness of a new agent (Ag_b). To simplify the problem, we supposed that each consulted agent (a *confidence agent*) offers a trustworthiness value of Ag_b if he knows him. If a confidence agent does not offer any trustworthiness value, it will not be taken into account at the moment of the evaluation of Ag_b's trustworthiness by Ag_a. However, a confidence agent can, if he does not know Ag_b, offer to Ag_a a set of agents who eventually know Ag_b. In this case, Ag_a will ask the proposed agents. These agents also have a trustworthiness value according to the point of view of the agent who proposed them. For this reason, Ag_a applies Equation 5 to assess the trustworthiness values of these agents. These new values will be used to evaluate the

Ag_b's trustworthiness. We can build a trust graph in order to deal with this issue. We define such a graph as follows:

Definition 3 (Trust Graph). *A trust graph is a directed and weighted graph. The nodes are agents and an edge* (Ag_i, Ag_j) *means that agent* Ag_i *knows agent* Ag_j*. The weight of the edge* (Ag_i, Ag_j) *is a pair* (x, y) *where x is the* Ag_j*'s trustworthiness according to the point of view of* Ag_i *and y is the interaction number between* Ag_i *and* Ag_j*. The weight of a node is the agent's trustworthiness according to the point of view of the source agent.*

According to this definition, in order to determine the trustworthiness of the target agent Ag_b, it is necessary to find the weight of the node representing this agent in the graph. The graph is constructed while Ag_a receives answers from the consulted agents. The evaluation process of the nodes starts when all the graph is built. This means that this process only starts when Ag_a has received all the answers from the consulted agents. The process terminates when the node representing Ag_b is evaluated. The graph construction and the node evaluation algorithms are given respectively by Algorithms 1 and 2.

Correctness of Algorithm 1: The construction of the trust graph is described as follows:

1- Agent Ag_a sends a request about the Ag_b's trustworthiness to all the confidence agents Ag_i. The nodes representing these agents (denoted *Node*(Ag_i)) are added to the graph. Since the trustworthiness values of these agents are known, the weights of these nodes (denoted *Weight*(*Node*(Ag_i))) can be evaluated. These weights are represented by $TRUST(Ag_i)_{Ag_a}$ (i.e. by Ag_i's trustworthiness according to the point of view of Ag_a).

2- Ag_a uses the primitive *Send*(Ag_i, *Investigation*(Ag_b)) in order to ask Ag_i to offer a trustworthiness value for Ag_b. The Ag_is' answers are recovered when they are offered in a variable denoted *Str* by *Str* = *Receive*(Ag_i). *Str.Agents* represents the set of agents referred by Ag_i. $Str.TRUST(Ag_j)_{Ag_i}$ is the trustworthiness value of an agent Ag_j (belonging to the set *Str.Agents*) from the point of view of the agent who referred him (i.e. Ag_i).

3- When a consulted agent answers by indicating a set of agents, these agents will also be consulted. They can be regarded as *potential witnesses*. These witnesses are added to a set called: *Potonial_Witnesses*. When a potential witness is consulted, he is removed from the set.

4- To ensure that the evaluation process terminates, two limits are used: the maximum number of agents to be consulted (*Limit_Nbr_Visited_Agents*) and the maximum number of witnesses who must offer an answer (*Limit_Nbr_Witnesses*). The variable *Nbr_Additional_Agents* is used to be sure that the first limit is respected when Ag_a starts to receive the answers of the consulted agents.

```
Construct-Graph($Ag_a$, $Ag_b$, Limit_Nbr_Visited_Agents, Limit_Nbr_Witnesses)
{
  Graph := $\emptyset$
  Nbr_Witnesses := 0
  Nbr_Visited_Agents := 0
  Nbr_Additional_Agents :=
                    Max(0, Limit_Nbr_Visited_Agents – Size(Confidence($Ag_a$)))
  Potential_Witnesses := Confidence($Ag_a$)
  Add Node($Ag_b$) to Graph

  While (Potential_Witnesses $\neq \emptyset$) and
        (Nbr_Witnesses < Limit_Nbr_Witnesses) and
        (Nbr_Visited_Agents < Limit_Nbr_Visited_Agents) {

      n := Limit_Nbr_Visited_Agents - Nbr_Visited_Agents
      m := Limit_Nbr_Witnesses - Nbr_Witnesses

      For (i =1, i $\leq$ min(n, m), i++) {
          $Ag_1$ := Potential_Witnesses(i)
          If Node($Ag_1$) $\notin$ Graph Then Add Node($Ag_1$) to Graph
          If $Ag_1 \in$ Confidence($Ag_a$) Then Weight(Node($Ag_1$)) := Trust($Ag_1$)$Ag_a$
          Send($Ag_1$, Investigation($Ag_b$))
          Nbr_Visited_Agents := Nbr_Visited_Agents +1 }

      For (i =1, i $\leq$ min(n, m), i++) {
          $Ag_1$ := Potential_Witnesses(1)
          Str := Receive($Ag_1$)
          Potential_Witnesses := Potential_Witnesses / {$Ag_1$}
          While (Str.Agents $\neq \emptyset$) and (Nbr_Additional_Agents > 0) {
              If Str.Agents = {$Ag_b$} Then {
                  Nbr_Witnesses := Nbr_Witnesses + 1
                  Add Arc($Ag_1$, $Ag_b$)
                  Weight1(Arc($Ag_1$, $Ag_b$)) := Str.TRUST$(Ag_b)_{Ag1}$
                  Weight2(Arc($Ag_1$, $Ag_b$)) := Str.n$(Ag_b)_{Ag1}$
                  Str.Agents := $\emptyset$ }
              Else {
                  Nbr_Additional_Agents := Nbr_Additional_Agents – 1
                  $Ag_2$ := Str.Agents(1)
                  Str.Agents := Str.Agents / {$Ag_2$}
                  If Node($Ag_2$) $\notin$ Graph then Add $Ag_2$ to Graph
                  Weight1(Arc($Ag_1$, $Ag_2$)) := Str.TRUST$(Ag_2)_{Ag1}$
                  Weight2(Arc($Ag_1$, $Ag_2$)) := Str.n$(Ag_2)_{Ag1}$
                  Potential_Witnesses := Potential_Witnesses $\cup$ {$Ag_2$} } } } }
}
```

Algorithm 1

```
Evaluate-Node($Ag_y$) {
    $\forall$Arc($Ag_x$, $Ag_y$)
        If Node($Ag_x$) is note evaluated Then
            Evaluate-Node($Ag_x$)

    m1 := 0, m2 := 0
    $\forall$Arc($Ag_x$, $Ag_y$) {
        m1 = m1 +
                Weight(Node($Ag_x$)) * Weight(Arc($Ag_x$, $Ag_y$))
        m2 = m2 + Weight(Node($Ag_x$))
    }
    Weight(Node($Ag_y$)) = m1 / m2
}
```

Algorithm 2

Correctness of Algorithm 2: The trustworthiness combination formula (Equation 5) is used to evaluate the graph nodes. The weight of each node indicates the trustworthiness value of the agent represented by the node. Such a weight is assessed using the weights of the adjacent nodes. For example, let *Arc*(Ag_x, Ag_y) be an arc in the graph, before evaluating Ag_y it is necessary to evaluate Ag_x. Consequently, the evaluation algorithm is recursive. The algorithm terminates because the nodes of the set *Confidence*(Ag_a) are already evaluated by Algorithm 1. Since the evaluation is done recursively, the call of this algorithm in the main program has as parameter the agent Ag_b.

Complexity Analysis. Our trustworthiness model is based on the construction of a trust graph and on a recursive call to the function *Evaluate-Node*(Ag_y) to assess the weight of all the nodes. Since each node is visited exactly once, there are *n* recursive calls, where *n* is the number of nodes in the graph. To assess the weight of a node we need the weights of its neighboring nodes and the weights of the input edges. Thus, the algorithm takes a time in $O(n)$ for the recursive calls and a time in $O(a)$ to assess the agents' trustworthiness where *a* is the number of edges. The run time of the trustworthiness algorithm is therefore in $O(max(a, n))$ i.e. linear in the size of the graph. Consequently, our algorithm is an efficient one.

4 Implementation

In this section we describe the implementation of our negotiation dialogue game framework and the trustworthiness model using the Jack^{TM} platform (The Agent Oriented Software Group, 2004). We select this language 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. A cursor is a representation of the results of a query. It is an enumerator which provides query result enumeration by means of re-binding the logical variables used in the query. 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.

4.1 General Architecture

Our system consists of two types of agents: *negotiating agents* and *trust model agents*. These agents are implemented as Jack^{TM} agents, i.e. they inherit from the basic class Jack^{TM} *Agent*. Negotiating agents are agents that take part in the negotiation protocol. Trust model agents are agents that can inform an agent about the trustworthiness of another agent (Fig. 3). Agents must have knowledge and argumentation systems. Agents' knowledge are implemented using Jack^{TM} data structures called *beliefsets*. The argumentation systems are implemented as Java modules using a logical programming paradigm. These modules use agents' beliefsets to build arguments for or against certain propositional formulae. The actions that agents perform on commitments or on their contents are programmed as *events*. When an agent receives such an event, it seeks a *plan* to handle it.

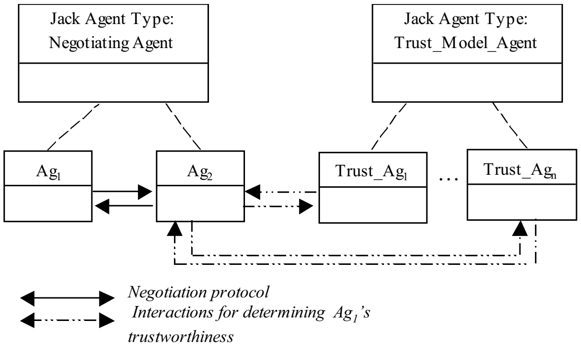
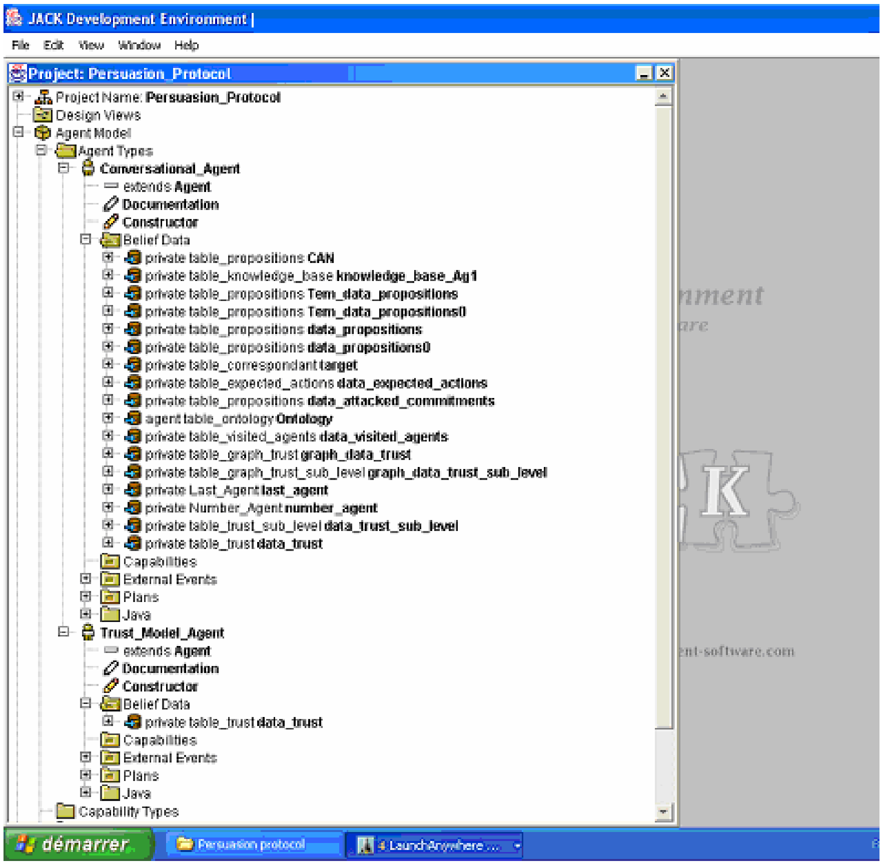


Fig. 3. The general architecture of the system

The trustworthiness model is implemented using the same principle (events + plans). The requests sent by an agent about the trustworthiness of another agent are events and the evaluations of agents' trustworthiness are programmed in plans. The trust graph is implemented as a Java data structure (oriented graph).

As Java classes, negotiating agents and trust model agents have private data called *Belief Data*. For example, the different commitments and arguments that are made and manipulated are given by a data structure called *CAN* implemented using tables and the different actions expected by an agent in the context of a particular negotiation game are given by a data structure (table) called *data_expected_actions*. The different agents' trustworthiness values that an agent has are recorded in a data structure (table) called *data_trust*. These data and their types are given in Fig. 4 and Fig. 5.

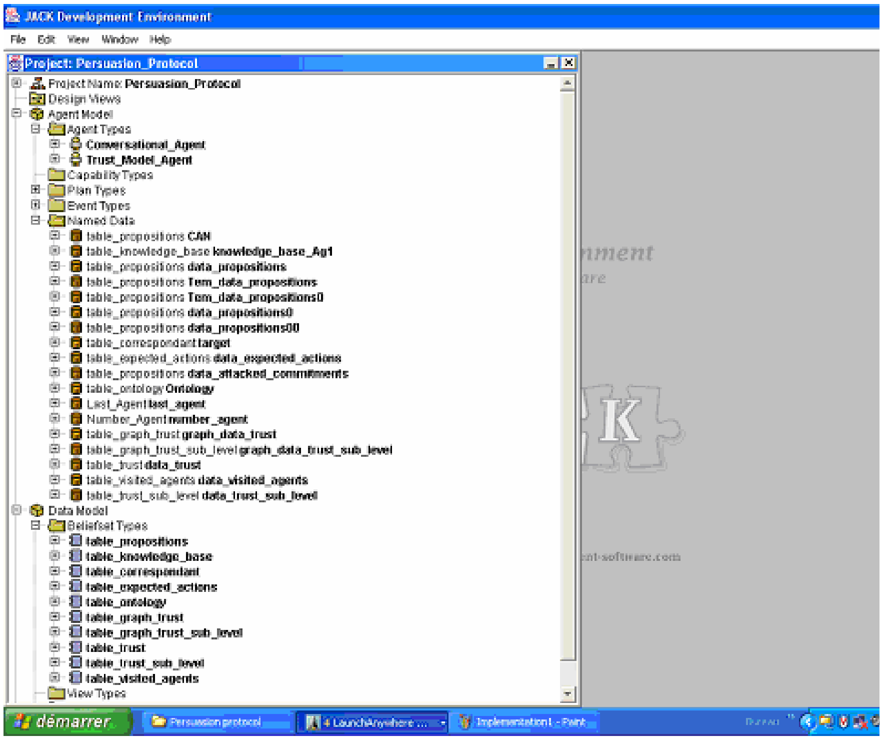


```
Project Name: Persuasion_Protocol
Design Views
Agent Model
    Agent Types
        Conversational_Agent
            extends Agent
            Documentation
            Constructor
            Belief Data
                private table_propositions CAN
                private table_knowledge_base knowledge_base_Ag1
                private table_propositions Tem_data_propositions
                private table_propositions Tem_data_propositions0
                private table_propositions data_propositions
                private table_propositions data_propositions0
                private table_correspondant target
                private table_expected_actions data_expected_actions
                private table_propositions data_attacked_commitments
                agent table_ontology Ontology
                private table_visited_agents data_visited_agents
                private table_graph_trust graph_data_trust
                private table_graph_trust_sub_level graph_data_trust_sub_level
                private Last_Agent last_agent
                private Number_Agent number_agent
                private table_trust_sub_level data_trust_sub_level
                private table_trust data_trust
            Capabilities
            External Events
            Plans
            Java
        Trust_Model_Agent
            extends Agent
            Documentation
            Constructor
            Belief Data
                private table_trust data_trust
            Capabilities
            External Events
            Plans
            Java
    Capability Types
```

Fig. 4. Belief Data used in our prototype

4.2 Implementation of the Trustworthiness Model

The trustworthiness model is implemented by agents of type: trust model agent. Each agent of this type has a knowledge base implemented using JackTM beliefsets. This knowledge base, called table_trust, has the following structure: Agent_name, Agent_trust, and Interaction_number. Thus, each agent has information on other agents about their trustworthiness and the number of times that he interacted with them. The visited agents during the evaluation process and the agents added in the trust graph are recorded in two JackTM beliefsets called: table_visited_agents and table_graph_trust. The two limits used in Algorithm 1 (Limit_Nbr_Visited_Agents and Limit_Nbr_Witnesses) and the trustworthiness threshold w are passed as parameters to the JackTM constructor of the original agent Aga that seeks to know if his interlocutor Agb is trustworthy or not. This original agent is a negotiating agent.


JACK Development Environment
File Edit View Window Help
Project: Persuasion_Protocol
démarrer
Persuasion protocol
4 LaunchAnywhere ...
Implementation1 - Paint

```
Project Name: Persuasion_Protocol
Design Views
Agent Model
    Agent Types
        Conversational_Agent
        Trust_Model_Agent
    Capability Types
    Plan Types
    Event Types
    Named Data
        table_propositions CAN
        table_knowledge_base knowledge_base_Ag1
        table_propositions data_propositions
        table_propositions Tem_data_propositions
        table_propositions Tem_data_propositions0
        table_propositions data_propositions0
        table_propositions data_propositions00
        table_correspondant target
        table_expected_actions data_expected_actions
        table_propositions data_attacked_commitments
        table_ontology Ontology
        Last_Agent last_agent
        Number_Agent number_agent
        table_graph_trust graph_data_trust
        table_graph_trust_sub_level graph_data_trust_sub_level
        table_trust data_trust
        table_visited_agents data_visited_agents
        table_trust_sub_level data_trust_sub_level
Data Model
    Beliefset Types
        table_propositions
        table_knowledge_base
        table_correspondant
        table_expected_actions
        table_ontology
        table_graph_trust
        table_graph_trust_sub_level
        table_trust
        table_trust_sub_level
        table_visited_agents
    View Types
```

Fig. 5. Beliefsets used in our prototype

The main steps of the evaluation process of Ag_b's trustworthiness are implemented as follows:

1- By respecting the two limits and the threshold w , Ag_a consults his knowledge base *data_trust* of type *table_trust* and sends a request to his confidence agents Ag_i (i = 1,.., n) about Ag_b's trustworthiness. The Jack^{TM} primitive *Send* makes it possible to send the request as a Jack^{TM} message that we call *Ask_Trust* of *MessageEvent* type. Ag_a sends this request starting by confidence agents whose trustworthiness value is highest.

2- In order to answer the Ag_a's request, each agent Ag_i executes a Jack^{TM} plan instance that we call *Plan_ev_Ask_Trust*. Thus, using his knowledge base, each agent Ag_i offers to Ag_a an Ag_b's trustworthiness value if Ag_b is known by Ag_i. If not, Ag_i proposes a set of confidence agents from his point of view, with their trustworthiness values and the number of times that he interacted with them. In the first case, Ag_i sends to Ag_a a Jack^{TM} message that we call *Trust_Value*. In the second case, Ag_i sends a message that we call *Confidence_Agent*. These two messages are of type *MessageEvent*.

3- When Ag_a receives the *Trust_Value* message, he executes a plan: *Plan_ev_Trust_Value*. According to this plan, Ag_a adds to a graph structure called *graph_data_trust* two information: 1) the agent Ag_i and his trustworthiness value as graph node; 2) the trustworthiness value that Ag_i offers for Ag_b and the number of times that Ag_i interacted with Ag_b as arc relating the node Ag_i and the node Ag_b. This first part of the trust graph is recorded until the end of the evaluation process of Ag_b's trustworthiness. When Ag_a receives the *Confidence_Agent* message, he executes another plan: *Plan_ev_Confidence_Agent*. According to this plan, Ag_a adds to another graph structure: *graph_data_trust_sub_level* three information for each Ag_i agent: 1) the agent Ag_i and his trustworthiness value as a sub-graph node; 2) the nodes Ag_j representing the agents proposed by Ag_i; 3) For each agent Ag_j, the trustworthiness value that Ag_i assigns to Ag_j and the number of times that Ag_i interacted with Ag_j as arc between Ag_i and Ag_j. This information that constitutes a sub-graph of the trust graph will be used to evaluate Ag_j's trustworthiness values using Equation 5. These values are recorded in a new structure: *new_data_trust*. Thus, the structure *graph_data_trust_sub_level* releases the memory once Ag_j's trustworthiness values are evaluated. This technique allows us to decrease the space complexity of our algorithm.

4- Steps 1, 2, and 3 are applied again by substituting *data_trust* by *new_data_trust*, until all the consulted agents offer a trustworthiness value for Ag_b or until one of the two limits (*Limit_Nbr_Visited_Agents* or *Limit_Nbr_Witnesses*) is reached.

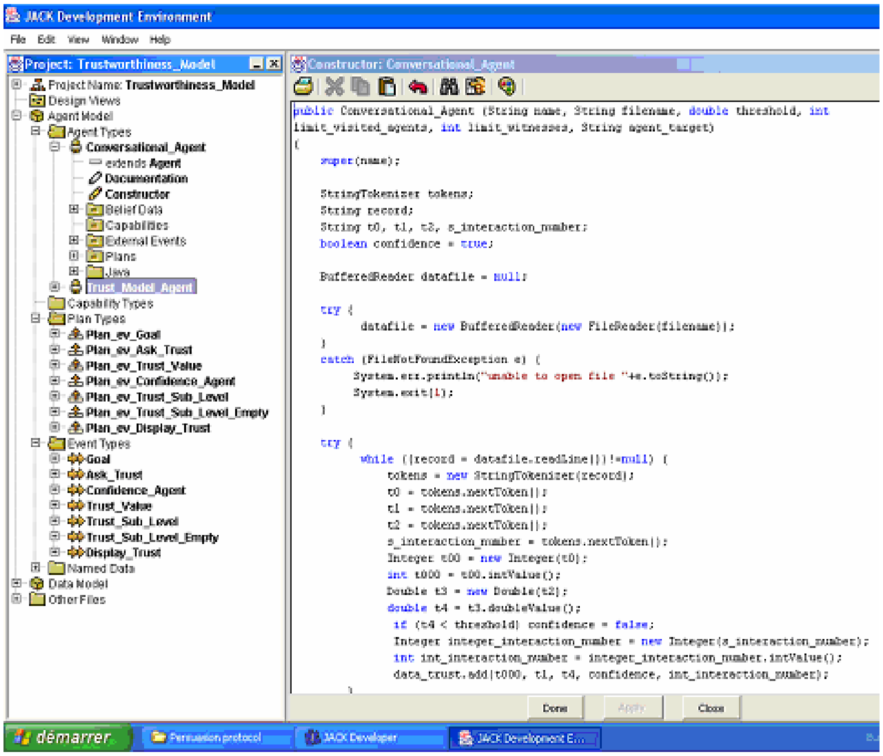
5- Evaluate the Ag_b's trustworthiness value using the information recorded in the structure *graph_data_trust* by applying Equation 5.

The different events and plans implementing our trustworthiness model and the negotiating agent constructor are illustrated by Fig. 6. Fig. 7 illustrates an example generated by our prototype of the process allowing an agent Ag_1 to assess the trustworthiness of another agent Ag_2. In this example, Ag_2 is considered trustworthy by Ag_1 because its trustworthiness value (0.79) is higher than the threshold (0.7).

4.3 Implementation of the Negotiation Dialogue Games

In our system, agents' knowledge bases contain propositional formulae and arguments. These knowledge bases are implemented as Jack^{TM} *beliefsets*. *Beliefsets* are used to maintain an agent's beliefs about the world. These beliefs are represented in a first order 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.

Our knowledge bases (KBs) contain two types of information: arguments and beliefs. Arguments have the form ([*Support*], *Conclusion*), where *Support* is a set of propositional formulae and *Conclusion* is a propositional formula. Beliefs have the form ([*Belief*], *Belief*) i.e. *Support* and *Conclusion* are identical. The meaning of the propositional formulae (i.e. the ontology) is recorded in a *beliefset* called *table_ontology* whose access is shared between the two agents. This beliefset has two fields: *Proposition* and *Meaning*.



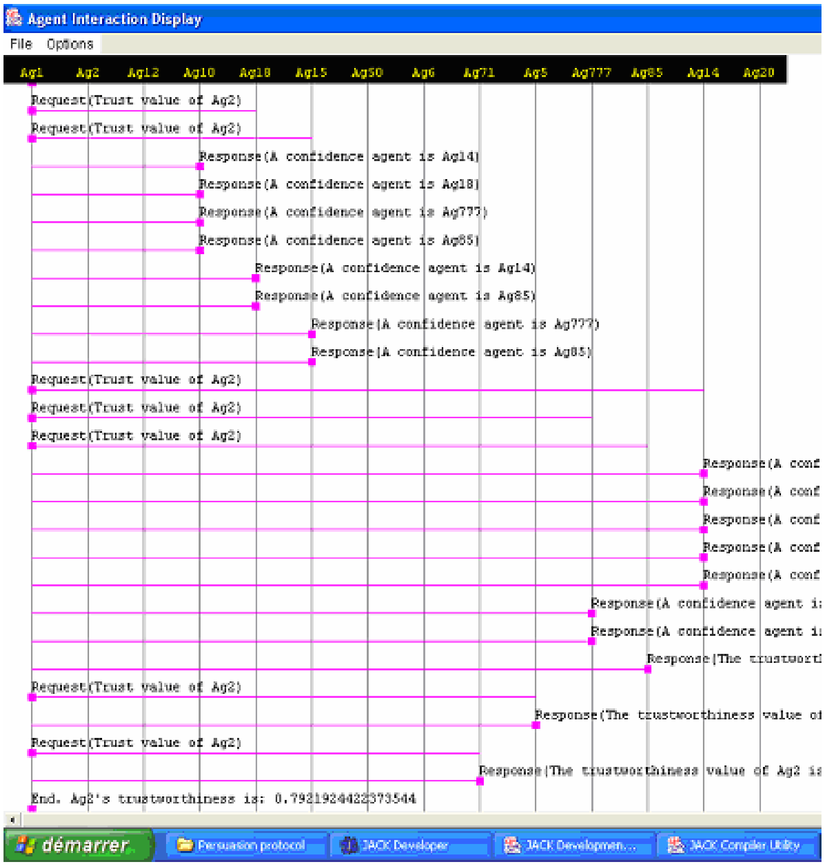
```
public Conversational_Agent (String name, String filename, double threshold, int
limit_visited_agents, int limit_witnesses, String agent_target)
{
    super(name);

    StringTokenizer tokens;
    String record;
    String t0, t1, t2, s_interaction_number;
    boolean confidence = true;

    BufferedReader datafile = null;

    try {
          datafile = new BufferedReader(new FileReader(filename));
    }
    catch (FileNotFoundException e) {
         System.err.println("unable to open file "+e.toString());
         System.exit(1);
    }

    try {
          while ((record = datafile.readLine())!=null) {
              tokens = new StringTokenizer(record);
              t0 = tokens.nextToken();
              t1 = tokens.nextToken();
              t2 = tokens.nextToken();
              s_interaction_number = tokens.nextToken();
              Integer t00 = new Integer(t0);
              int t000 = t00.intValue();
              Double t3 = new Double(t2);
              double t4 = t3.doubleValue();
               if (t4 < threshold) confidence = false;
               Integer integer_interaction_number = new Integer(s_interaction_number);
               int int_interaction_number = integer_interaction_number.intValue();
               data_trust.add(t000, t1, t4, confidence, int_interaction_number);
          }
```

Fig. 6. Events, plans and the conversational agent constructor implementing the trustworthiness model

Agent communication is done by sending and receiving messages. These messages are *events* that extend the basic Jack^{TM} *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*).

Our negotiation dialogue games are implemented as a set of *events* (*MessageEvents*) 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 reasoning 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 his interlocutor 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. For example, the event *Event_Attack_Commitment* and the plan *Plan_ev_Attack_commitment* implement the Attack game. The architecture of our negotiating agents is illustrated in Fig. 8.



```
Request(Trust value of Ag2)
Request(Trust value of Ag2)
Response(A confidence agent is Ag14)
Response(A confidence agent is Ag18)
Response(A confidence agent is Ag777)
Response(A confidence agent is Ag85)
Response(A confidence agent is Ag14)
Response(A confidence agent is Ag85)
Response(A confidence agent is Ag777)
Response(A confidence agent is Ag85)
Request(Trust value of Ag2)
Request(Trust value of Ag2)
Request(Trust value of Ag2)
Response(A conf
Response(A conf
Response(A conf
Response(A conf
Response(A conf
Response(A confidence agent i
Response(A confidence agent i
Response(The trustwort
Request(Trust value of Ag2)
Response(The trustworthiness value o
Request(Trust value of Ag2)
Response(The trustworthiness value of Ag2 i
End. Ag2's trustworthiness is: 0.792192442237354
```

Fig. 7. The screen shot of a trustworthiness evaluation process

5 Related Work

Recently, some online trust models have been developed (see [20] for a detailed survey). The most widely used are those on eBay and Amazon Auctions. Both of these are implemented as a centralized trust system so that their users can rate and learn about each other's reputation. For example, on eBay, trust values (or ratings) are +1, 0, or –1 and user, after an interaction, can rate its partner. The ratings are stored centrally and summed up to give an overall rating. Thus, reputation in these models is a global single value. However, the model can be unreliable, particularly when some buyers do not return ratings. In addition, these models are not suitable for applications in open MAS such as agent negotiation because they are too simple in terms of their trust rating values and the way they are aggregated.

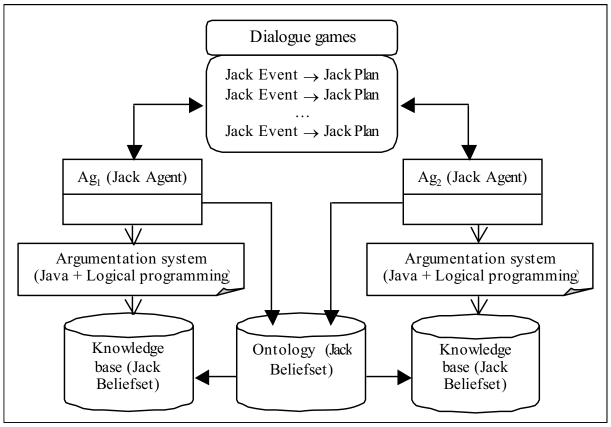

Dialogue games
Jack Event → Jack Plan
Jack Event → Jack Plan
…
Jack Event → Jack Plan
Ag1 (Jack Agent)
Ag2 (Jack Agent)
Argumentation system (Java + Logical programming)
Argumentation system (Java + Logical programming)
Knowledge base (Jack Beliefset)
Ontology (Jack Beliefset)
Knowledge base (Jack Beliefset)

Fig. 8. The architecture of the negotiating agents

Another centralized approach called SPORAS has been proposed by Zacharia and Maes [7]. SPORAS does not store all the trust values, but rather updates the global reputation value of an agent according to its most recent rating. The model uses a learning function for the updating process so that the reputation value can reflect an agent's trust. In addition, it introduces a reliability measure based on the standard deviations of the trust values. However, unlike our models, SPORAS deal with all ratings equally without considering the different trust degrees. Consequently, it suffers from rating noise. In addition, like eBay, SPORAS is a centralized approach, so it is not suitable for open negotiation systems.

Broadly speaking, there are three main approaches to trust in open multi-agent systems. The first approach is built on an agent's direct experience of an interaction partner. The second approach uses information provided by other agents [2, 3, 4]. The third approach uses certified information provided by referees [9, 19]. In the first approach, methods by which agents can learn and make decisions to deal with trustworthy or untrustworthy agents should be considered. In the models based on the second and the third approaches, agents should be able to reliably acquire and reason about the transmitted information. In the third approach, agents should provide third-party referees to witness about their previous performance. Because the first approaches are only based on a history of interactions, the resulting models are poor because agents with no prior interaction histories could trust dishonest gents until a sufficient number of interactions is built.

Sabater [13] proposes a decentralized trust model called Regret. Unlike the first approach models, Regret uses an evaluation technique not only based on an agent's direct experience of its partners reliability, but it also uses a witness reputation component. In addition, trust values (called ratings) are dealt with according to their recency relevance. Thus, old ratings are given less importance compared to new ones.

However, unlike our model, Regret does not show how witnesses can be located, and thus, this component is of limited use. In addition, this model does not deal with the possibility that an agent may lie about its rating of another agent, and because the ratings are simply equally summed, the technique can be sensitive to noise. In our model, this issue is managed by considering the witnesses' trust and because our merging method takes into account the proportional relevance of each trustworthiness value, rather than treating them equally (see Equation 6 Section III.B)

Yu and Singh [2, 3, 4] propose an approach based on social networks in which agents, acting as witnesses, can transmit information about each other. The purpose is to tackle the problem of retrieving ratings from a social network through the use of *referrals*. Referrals are pointers to other sources of information similar to links that a search engine would plough through to obtain a Web page. Through referrals, an agent can provide another agent with alternative sources of information about a potential interaction partner. The social network is presented using a referral network called *TrustNet*. The trust graph we propose in this paper is similar to TrustNet, however there are several differences between our approach and Yu and Singh's approach. Unlike Yu and Singh's approach in which agents do not use any particular reasoning, our approach is conceived to secure argumentation-based negotiation in which agents use an argumentation-based reasoning. In addition, Yu and Singh do not consider the possibility that an agent may lie about its rating of another agent. They assume all witnesses are totally honest. However, this problem of inaccurate reports is considered in our approach by taking into account the trust of all the agents in the trust graph, particularly the witnesses. Also, unlike our model, Yu and Singh's model do not treat the timely relevance information and all ratings are dealt with equally. Consequently, this approach cannot manage the situation where the agents' behavior changes.

Huynh, Jennings, and Shadbot [19] tackle the problem of collecting the required information by the evaluator itself to assess the trust of its partner, called the target. The problem is due to the fact that the models based on witness implicitly assume that witnesses are willing to share their experiences. For this reason, they propose an approach, called *certified reputation*, based not only on direct and indirect experiences, but also on third-party references provided by the target agent itself. The idea is that the target agent can present arguments about its reputation. These arguments are references produced by the agents that have interacted with the target agents certifying its credibility (the model proposed by Maximilien and Singh [5] uses the same idea). This approach has the advantage of quickly producing an assessment of the target's trust because it only needs a small number of interactions and it does not require the construction of a trust graph. However, this approach has some serious limitations. Because the referees are proposed by the target agent, this agent can provide only referees that will give positive ratings about it and avoid other referees, probably more credible than the provided ones. Even if the provided agents are credible, their witness could not reflect the real picture of the target's honesty. This approach can privilege *opportunistic agents*, which are agents only credible with potential referees. For all these reasons, this approach is not suitable for trusting negotiating agents. In addition, in this approach, the evaluator agent should be able to evaluate the honesty of the referees using a witness-based model. Consequently, a trust graph like the one proposed in this paper could be used. This means that, in some situations, the target's trust might not be assessed without asking for witness agents.

6 Conclusion

The contribution of this paper is the proposition and the implementation of a new probabilistic model to trust argumentation-based negotiating agents. The purpose of such a model is to provide a secure environment for agent negotiation within multi-agent systems. To our knowledge, this paper is the first work addressing the security issue of argumentation-based negotiation in multi-agent settings. Our model has the advantage of being computationally efficient and of gathering four most important factors: (1) the trustworthiness of confidence agents; (2) the target's trustworthiness according to the point of view of confidence agents; (3) the number of interactions between confidence agents and the target agent; and (4) the timely relevance of information transmitted by confidence agents. The resulting model allows us to produce a comprehensive assessment of the agents' credibility in an argumentation-based negotiation setting.

Acknowledgements

We would like to thank the Natural Sciences and Engineering Research Council of Canada (**NSERC**), le fonds québécois de la recherche sur la nature et les technologies (**NATEQ**), and le fonds québécois de la recherche sur la société et la culture (**FQRSC**) for their financial support. The first author is also supported in part by Concordia University, Faculty of Engineering and Computer Science (**Start-up Grant**). Also, we would like to thank the three anonymous reviewers for their interesting comments and suggestions.

References

[1] A. Abdul-Rahman, and S. Hailes. Supporting trust in virtual communities. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*, 6, IEEE Computer Society Press. 2000.

[2] B. Yu, and M. P. Singh. An evidential model of distributed reputation management. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*. ACM Press, pages 294–301, 2002.

[3] B. Yu, and M. P. Singh. Detecting deception in reputation management. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multi-Agent Systems*. ACM Press, pages 73-80, 2003.

[4] B. Yu, and M. P. Singh. Searching social networks. In *Proceedings of the second International Joint Conference on Autonomous Agents and Multi-Agent Systems*. ACM Press, pp. 65–72, 2003.

[5] E. M. Maximilien, and M. P. Singh. Reputation and endorsement for web services. *ACM SIGEcom Exchanges*, 3(1):24-31, 2002.

[6] F. Sadri, F. Toni, and P. Torroni. Dialogues for negotiation: agent varieties and dialogue sequences. In *Proceedings of the International workshop on Agents, Theories, Architectures and Languages*. Lecture Notes in Artificial Intelligence (2333):405–421, 2001.

[7] G. Zacharia, and P. Maes. Trust management through reputation mechanisms. *Applied Artificial Intelligence*, 14(9):881-908, 2000.

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

[9] H. Skogsrud, B. Benatallah, and F. Casati. Model-driven trust negotiation for web services. *IEEE Internet Computing*, 7(6):45-52, 2003.

[10] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: enabling the scalable virtual organization. *The International Journal of High Performance Computing Applications*, 15(3), 200-222, 2001.

[11] J. Bentahar, B. Moulin, J-.J. Ch. Meyer, and B. Chaib-draa. A computational model for conversation policies for agent communication. In J. Leite and P. Torroni editors, *Computational Logic in Multi-Agent Systems*. Lecture Notes in Artificial Intelligence (3487): 178-195, 2005.

[12] J. Bentahar. *A pragmatic and semantic unified framework for agent communication*. Ph.D. Thesis, Laval University, Canada, May 2005.

[13] J. Sabater. *Trust and Reputation for Agent Societies*. Ph.D. Thesis, Universitat Autμonoma de Barcelona, 2003.

[14] L. Amgoud, N. Maudet, S. Parsons. Modelling dialogues using argumentation. In *Proceeding of the 4th International Conference on Multi-Agent Systems*, pages 31-38, 2000.

[15] M. Dastani, J. Hulstijn, and L. V. der Torre. Negotiation protocols and dialogue games. In *Proceedings of Belgium/Dutch Artificial Intelligence Conference*, pages 13-20, 2000.

[16] N. Maudet, and B. Chaib-draa, Commitment-based and dialogue-game based protocols, new trends in agent communication languages. In *Knowledge Engineering Review*. Cambridge University Press, 17(2):157-179, 2002.

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

[18] S. D. Ramchurn, T. D. Huynh, and N. R. Jennings. Trust in multi-agent systems. *The Knowledge Engineering Review*, 19(1):1-25, March 2004.

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

[20] T. Grandison, and M. Sloman. A survey of trust in internet applications. *IEEE Communication Surveys & Tutorials*, 3(4), 2000.