Securing Agent-Oriented Systems: An Argumentation and Reputation-based Approach

Jamal Bentahar
Concordia University, Concordia Institute for Information Systems Engineering, Canada
bentahar@ciise.concordia.ca

John-Jules Ch. Meyer
Utrecht University, Department of Information & Computer Science, The Netherlands
jj@cs.uu.nl

Bernard Moulin
Laval University, Department of Computer Science & Software Eng., Canada
bernard.moulin@ift.ulaval.ca

Abstract

In this paper, we propose a new model for securing agent-based systems in which agents are equipped with reasoning capabilities allowing them to interact with each other. The agents can reason about the reputation of each other using their argumentation systems. The reputation is dealt with as a quantitative value computed using a set of parameters based on the interaction histories and the notion of social networks. The problem of securing autonomous interacting agents 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 fail to adequately address the challenges of security 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 agent systems in which agents use advanced reasoning capabilities.

1. Introduction

Software systems involving autonomous interacting agents present new challenges in software engineering. A particularly challenging problem is the security engineering [16]. The problem of securing these systems 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 [6,17].

The objective of this paper is to address this challenging issue by proposing a new efficient computational model to secure agent-oriented systems using a reputation-based approach. The idea is to be able 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. Current approaches to security fail to adequately address the e-computing challenges. 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 [8,14,18,19,20,21] (see [13] for a survey). However, these models are not suitable for argumentation-based agents, 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. Moreover, unlike the agent-based approach proposed in [16], the issues addressed here are not about authentication and authorization but about trusting agents in a multi-agent setting.

International Conference on Information Technology (ITNG'07)
0-7695-2776-0/07 $20.00

At the architectural level of our approach, an agent-oriented system is abstracted as a society of agents interacting with each other using advanced communication techniques based on dialogue games [5,9,11] and argumentation reasoning [3,15]. Dialogue games are interactions between players (agents), in which each player moves by performing utterances according to a pre-defined set of roles. Argumentation can be abstractly defined as a dialectical process for the interaction of different arguments for and against some conclusion [2]. Agents can be assisted by argumentation to reach a decision, to inform, persuade, or negotiate with other agents. Moreover, 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. Unlike classical agent communication protocols, argumentation-based dialogue game protocols are flexible and more suitable for agents equipped with reasoning capabilities. 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 using their argumentation capabilities [3].

The rest of this paper is organized as follows. In Section 2, we present the main ideas of the interaction framework agents use in their interactions within a multi-agent system. In Section 3, we present our reputation-based model to secure agent-oriented systems. 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 we conclude.

2. Interaction Framework

In this section, we present the main ideas of the dialogue game-based framework for agent communication. In this framework, agents use their reasoning and argumentation abilities in order to decide about the next communicative act to perform. The framework is based on formal dialectics in which arguments are used as a way of expressing decision-making in terms of the dialogue games to be played [2,3,4,12]. The game moves are considered as actions that agents apply to the communicative acts (accept, refuse, challenge, question, justify, attack, etc.). This action is denoted $Action\_Ag_i$ where Ag_i is the actor. A dialogue game is specified as follows:

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

This specification indicates that if an agent Ag_1 performs the action $Action\_Ag_1$, and that the condition *Cond* is satisfied, then the interlocutor Ag_2 will perform the action $Action\_Ag_2$. The condition *Cond* is expressed in terms of the possibility of generating an argument from the agent's argumentation system supporting the action to be performed.

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* $h_1 \equiv \neg h_2$.

The combination of different dialogue games is specified using a set of logical rules. For example, according to a logical rule, before informing an agent that the proposition h is true, 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 it. On the other side, the addressee agent must use his own argumentation system to select the answer he will give (Accept, Challenge, etc.).

3. Reputation Model

3.1. Foundations

In recent years, several models of reputation and trust have been developed in the context of MAS [8,13,14,18,19,20]. However, these models are not designed to trust argumentation-based interacting agents. 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 agent-based systems that overcome some limitations of these models.

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

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

This function associates to each agent a probability measure representing its reputation in the domain D according to another agent. Let X be a random variable representing an agent's reputation. To evaluate the reputation of an agent Ag_b, an agent Ag_a uses the history of its interactions with Ag_b. Equation 1 indicates how to assess this reputation as a probability measure (*number of successful outcomes / total number of possible outcomes*).

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

$Rep(Ag_b)_{Ag_a}$ indicates the reputation 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_b*s' 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 reputation degree of an agent can be induced according to how much information acquired from him has been accepted as belief in the past. However, if the agent is new in the system, or if the number of interactions is not sufficient to determine this reputation, the consultation of other agents becomes necessary.

As proposed in [1,18,19], each agent has two kinds of beliefs when evaluating the reputation 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. Assessing Agent's Reputation

Let us suppose that an agent Ag_a wants to evaluate the reputation 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 reputation threshold w must be fixed. Thus, Ag_b will be considered trustworthy by Ag_a iff $Rep(Ag_b)_{Ag_a}$ is higher or equal to w. Ag_a attributes a reputation measure to each confidence agent Ag_i. When he is consulted by Ag_a, each confidence agent Ag_i provides a reputation 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 reputation using the reputation values transmitted by confidence agents.

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. Probability values offered by confidence agents are not mutually exclusive since they are provided simultaneously.

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

$$E(X) = p \quad (2)$$

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 $Rep(Ag_b)_{Ag_a}$. 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 ($X_1 + \ldots + X_n$)/n will be approximately normally distributed with mean μ. As an application of this theorem, the arithmetic mean (average) ($X_1 + \ldots + X_n$)/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 reputation 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_i 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 estimation of expectation $E(X)$ can be given by Equation 3.

$$M_0 = \frac{\sum_{i=1}^{n} Rep(Ag_i)_{Ag_a} Rep(Ag_b)_{Ag_i}}{\sum_{i=1}^{n} Rep(Ag_i)_{Ag_a}} \quad (3)$$

The value M_0 represents a first estimation of $Rep(Ag_b)_{Ag_a}$. This estimation, however, 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. 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 a richer estimation of $Rep(Ag_b)_{Ag_a}$ if we take into account these factors and we suppose that all confidence agents have the same reputation.

$$M_1 = \frac{\sum_{i=1}^{n} N(Ag_i)_{Ag_b} TR(Ag_i)_{Ag_b} Rep(Ag_b)_{Ag_i}}{\sum_{i=1}^{n} N(Ag_i)_{Ag_b} TR(Ag_i)_{Ag_b}} \quad (4)$$

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 reputation (*TR* denotes timely relevance). This factor is assessed by using the function defined in Equation 5. We call this function: the timely relevance function.

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

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. Consequently, the more recent the information is, the higher is the timely relevance coefficient (Fig. 1). 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 ($R(t) = e^{-\lambda t}$).

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

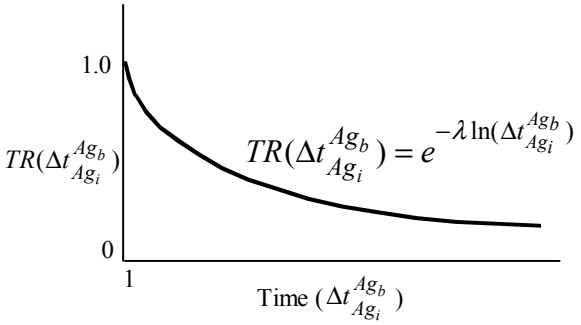

Fig. 1. The timely relevance function

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

This Equation shows how reputation can be obtained by merging the reputation values transmitted by some mediators. This merging method takes into account the proportional relevance of each reputation value, rather than treating them equally. To assess *M*, we need the reputation of other agents. To deal with this issue, we propose the notion of *reputation graph*.

3.3. Reputation Graph

In the previous section, we provided a solution to the reputation combination problem to evaluate the reputation of a new agent (Ag_b). To simplify the problem, we supposed that each consulted agent (a *confidence agent*) offers a reputation value of Ag_b if he

knows him. If a confidence agent does not offer any value, it will not be taken into account at the moment of the evaluation of Ag_b's reputation 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 reputation value according to the point of view of the agent who proposed them. For this reason, Ag_a applies Equation 5 to assess the reputation values of these agents. These new values will be used to evaluate the Ag_b's reputation. We can build a reputation graph in order to deal with this issue. We define such a graph as follows:

Definition 3 (Reputation Graph). *A reputation 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 reputation 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 reputation according to the point of view of the source agent.*

According to this definition, in order to determine the reputation 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 the entire 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 termination is guaranteed since the number of consulted agents is finite. The graph construction algorithm is as follows:

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

2- Ag_a asks Ag_i to offer a reputation value for Ag_b. The Ag_is' answers are recovered when they are offered in a variable denoted *Str*. *Str.Agents* represents the set of agents referred by Ag_i. $Str.Rep(Ag_j)_{Ag_i}$ is the reputation 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 reputation combination formula (Equation 5) is used to evaluate the graph nodes. The weight of each node indicates the reputation 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. This algorithm terminates because the nodes of the set $Confidence(Ag_a)$ are already evaluated by the construction graph algorithm. Since the evaluation is done recursively, the call of this algorithm in the main program has as parameter the agent Ag_b. Fig. 2 illustrates the node evaluation algorithm.

Complexity. Our reputation model is based on the construction of a trust graph and on a recursive call to the function $Evaluate\text{-}Node(Ag_y)$ 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' reputation where a is the number of edges. The run time of the reputation algorithm is therefore in $O(max(a, n))$ i.e. linear in the size of the graph.

4. Development

In this section we describe the implementation of the reputation model using the Jack™ platform (The Agent-Oriented Software Group, 2004). We have used this language for three main reasons: **1)** it is an agent-oriented language offering a framework for multi-agent system development which can support different agent models; **2)** it is built on top of and fully integrated with the Java programming language 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 with

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

    m1 := 0, m2 := 0
    ∀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
}
```

Fig. 2. The node evaluation algorithm

the addition of type checking Java style and embedded SQL.

4.1. Architecture and Implementation

The system is designed as a society of interacting agents equipped with knowledge bases and argumentation systems. Agents' knowledge bases contain propositional formulae and arguments. These knowledge bases are designed and implemented as Jack™ data structures called *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*.

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*).

As explained in Section 2, agents communicate using a set of dialogue games. These 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 interacting 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* and the plan *Plan_ev_Attack* implement the Attack game. The system architecture is illustrated in Fig. 3.

4.2. Prototype

To assess our reputation model, we have implemented a prototype using the architecture described above. In this prototype, agents are implemented as Jack™ agents, i.e. they inherit from the basic class Jack™ *Agent*. The argumentation systems are implemented as Java modules using a logical programming paradigm. These modules use

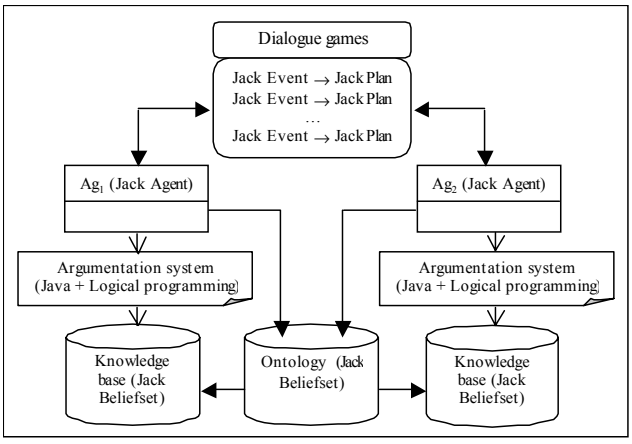

Fig. 3. The system architecture

agents' *beliefsets* to build arguments for or against certain propositional formulae. The reputation model is implemented using events and plans. The requests sent by an agent about the reputation of another agent are events and the evaluations of agents' reputation are programmed in plans. The reputation graph is implemented as a Java data structure (oriented graph).

As Java classes, 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 dialogue game are given by a table called *data_expected_actions*. The different agents' reputation values that an agent has are recorded in a data structure (table) called *data_reputation*.

Each agent has also a knowledge base about the reputation of other agents, called *table_reputation*. Implemented using Jack™ *beliefsets*, this knowledge base has the following structure: *Agent-name*, *Agen-reputation*, *Interaction-number*, and *Interaction-period*. Thus, each agent has information about other agents' reputation, the number of times he interacted with them, and the period during which they have interacted. The visited agents during the evaluation process and the agents added in the reputation graph are recorded in two Jack™ *beliefsets* called: *table_visited_agents* and *table_graph_reputation*. The two limits used in the graph construction algorithm and the reputation threshold are parameters of the Jack™ constructor of the initiator agent Ag_a that seeks to know if his interlocutor Ag_b is trustworthy or not.

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

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

2- In order to answer the Ag_a's request, each agent Ag_i executes a plan instance that we call *Plan_ev_Ask_Reputation*. Thus, using his knowledge base, each agent Ag_i offers to Ag_a an Ag_b's reputation 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 reputation values, the number of times that he interacted with them, and the time difference Δt.

3- When Ag_a receives the *Reputation_Value* message, he executes a plan: *Plan_ev_Reputation_Value*. According to this plan, Ag_a adds to a graph structure called *graph_data_reputation* two information: 1) the agent Ag_i and his reputation value as graph node; 2) The reputation value that Ag_i offers for Ag_b, the number of times that Ag_i interacted with Ag_b , and the time difference Δt as arc relating the node Ag_i and the node Ag_b. This first part of the reputation graph is recorded until the end of the evaluation process of Ag_b's reputation. 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_reputation_sub_level* three information for each Ag_i agent: 1) the agent Ag_i and his reputation 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 reputation value that Ag_i assigns to Ag_j, the number of times that Ag_i interacted with Ag_j, and the time difference Δt as arc between Ag_i and Ag_j. This information that constitutes a sub-graph of the reputation graph will be used to evaluate Ag_j's reputation values using Equation 5. These values are recorded in a new structure: *new_data_reputation*. Thus, the structure *graph_data_reputation_sub_level* releases the memory once Ag_j's reputation 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_reputation* by *new_data_reputation*, until all the consulted agents offer a reputation value for Ag_b or until one of the two fixed limits is reached.

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

5. Performance Evaluation

Recently, some online trust models have been developed (see [7] 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.

Another centralized approach called SPORAS has been proposed in [21]. 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. 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 [18,19,20]. The third approach uses certified information provided by referees [8,17]. 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 [14] 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 reputation value, rather than treating them equally (see Equation 6 Section 3.2)

Yu and Singh [18,19,20] 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 [8] 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 [10] 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 secure agent-oriented systems. To our knowledge, this paper is the first work addressing the security issue of argumentation-based agents in multi-agent settings. Our model has the advantage of being computationally efficient and of gathering four most important factors: (1) the reputation of confidence agents; (2) the target's reputation 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 software system.

Acknowledgements

The first author is supported in part by the Faculty of Engineering & Computer Science at Concordia University (start-up grant). This work is also supported by NSERC and FQRSC (Canada). We would also like to thank the anonymous reviewers for their interesting comments and suggestions.

7. References

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

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

[3] 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.

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

[5] M. Dastani, J. Hulstijn, and L. V. der Torre. Negotiation protocols and dialogue games. In *Proc. of Belgium/Dutch Artificial Intelligence Conf.*, pp. 13-20, 2000.

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

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

[8] 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.

[9] 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.

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

[11] 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.

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

[13] 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.

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

[15] 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.

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

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

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

[19] B. Yu, and M. P. Singh. Detecting deception in reputation management. In *Proc. of the 2nd Int. Conf. on AAMAS*. ACM Press, pp. 73-80, 2003.

[20] B. Yu, and M. P. Singh. Searching social networks. In *Proc. of the second Int. Conf. on AAMAS*, pp. 65–72, 2003.

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