Hybrid Reasoning and Coordination Methods on Multi-Agent Systems

S. Heras, M. Navarro and V. Julián

***Abstract*—This paper briefly introduces a summary of the special session on Hybrid Reasoning and Coordination Methods on Multi-Agent Systems, held in conjunction with the 4th International Conference on Hybrid Artificial Intelligence Systems 2009 (HAIS'09). The research papers of this session have been revised and extended and the final results are published in this journal special issue.**

***Index Terms*—Hybrid Multi-Agent Systems, Reasoning and Coordination Methods**

I. Introduction

Over the last years, Multi-Agent Systems (MAS) have been successfully applied to manage complex distributed processes in a wide range of application domains. In these systems, agents must be able to reason autonomously and coordinate their activities with other agents of the system to fulfil their objectives. Since the consolidation of the MAS paradigm in the 90's, much research has been done to provide social agents with better reasoning and coordination mechanisms that enhance their intelligence and improve their performance. Recently, the research community in MAS has focused their efforts on adapting MAS to open environments where heterogeneous agents interact. In these systems, agents can enter in (or leave) the system, form societies and communicate with other agents. Common assumptions about the agents of most MAS, such as honesty, cooperativeness and trustworthiness cannot be longer taken as valid hypothesis in open MAS. Therefore, the high dynamism of these open MAS gives rise to a greater need for complex reasoning and coordination mechanisms to control the access to the system and the deliberative processes of the agents.

The methods applied are very varied and the synergies with different areas of the Artificial Intelligence and other sciences have given rise to excellent results. Hybrid Artificial Intelligence Systems (HAIS) combine symbolic and sub-symbolic techniques to provide hybrid problem solving models. Their capabilities in handling many real world complex problems, involving imprecision, uncertainty and high-dimensionality makes them very suitable to cope with reasoning and coordination problems in complex open MAS.

The special session on Hybrid Multi-Agent Systems, Reasoning and Coordination Methods was aimed at discussing research on HAIS to develop reasoning and coordination methods on MAS. The session was conceived as a forum to present theoretical advances and real-world applications in this multidisciplinary research field.

Stella Heras, Martí Navarro and Vicente Julián
are with Universidad Politécnica de Valencia.
S. Heras: sheras@dsic.upv.es
M. Navarro: mnavarro@dsic.upv.es
V. Julián: vinglada@dsic.upv.es

II. Special Issue on Hybrid Multi-Agent Systems, Reasoning and Coordination Methods

This volume presents a revised version of the best papers presented in the special session on Hybrid Multi-Agent Systems, Reasoning and Coordination Methods, held in conjunction with the 4th International Conference on Hybrid Artificial Intelligence Systems 2009 (HAIS'09). Section encouraged topics, such as: practical reasoning methods, multi-agent case-based reasoning, artificial social systems, trust and reputation, social and organizational structure, teamwork, coalition formation, distributed problem solving, electronic markets and institutions, cooperative and non-cooperative game theory methods, social choice theory, voting protocols, auction and mechanism design, argumentation, negotiation and bargaining, agent commitments, semantic alignment and ontologies were tackled by the session papers, advancing research on the area of hybrid artificial intelligence systems. Each paper was reviewed by two independent experts in this area.

In the first paper, Muñoz and Botía propose an Argumentation System Based on Ontologies (ASBO) to cope with conflicts based on inconsistent knowledge which arise when agents exchange information. Their formal model follows an engineering-oriented approach to develop a software architecture which allows working with argumentation in MAS.

In the second paper, Pinzón et al. presents the core component of a solution based on agent technology that allows the identification of denial of service (DoS) attacks introduced in the XML of SOAP messages. The paper presents an advanced classification mechanism designed in two phases that are incorporated within a CBR-BDI Agent type. In addition, the method involves the use of decision trees, fuzzy logic rules and neural networks for filtering attacks. As a result of this work, a prototype was developed and the conclusions obtained are presented in the paper.

In the third paper, Búrdalo et al. propose a general tracing system that could be used by agents in the system to trace other agents' activity. This provides agents with an alternative way for perceiving their environment. The paper presents preliminary results of the authors' work, consisting of the requirements which should be taken into account when designing such a tracing system.

In the fourth paper, Castillo et al. show their work on developing a Multi-Agent Recommendation System (RecMAS) able to coordinate the interactions between a user agent and a set of commercial agents. The system provides a useful service

for monitoring changes in the user agents beliefs and decisions based on two parameters: (i) the strength of its own beliefs and (ii) the strength of the commercial agents suggestions. The system was used to test a prototype that copes with several commercial activities in a real shopping centre by using wireless devices (PDA, mobile phone, etc.). Using a theoretical model and the simulation experiments, commercial strategies in relation with the socio-dynamics of the system were obtained.

In the fifth paper, Heras et al. propose a new dialogue game protocol for modelling the interactions produced between the agents of an open MAS that must reach an agreement on the use of norms. The protocol is formally specified and the decision-making process of the agents is also developed. In addition, the authors provide an application example for showing both the performance of the protocol and its usefulness as a mechanism for managing the solving process of a coordination problem through norms.

Finally, in the last paper Navarro et al. copes with the problem of merging intelligent deliberative techniques with real-time reactive actions in the special context of Real-Time Multi-Agent Systems (RTMAS). In these systems, the temporal restrictions of their Real-Time Agents make their deliberation process to be temporally bounded. The paper proposes a solution based on a temporally bounded Case-Based Reasoning mechanism. Thus, it also presents a guide to adapt the Case-Based Reasoning cycle to be used as deliberative mechanism for Real-Time Agents.

Special Issue Acknowledgements

We would like to thank authors for their valuable work and their interest in the special session on Hybrid Multi-Agent Systems, Reasoning and Coordination Methods. Their contributions show high quality research in the area and their presentations gave rise to interesting discussions.

This special session could not be held without the support of the organising committee of HAIS 2009, to whom we are very grateful for their invaluable help. We also want to thank the experts of the program committee for their independent reviews and comments to the authors.

Finally, the Guest Editors wish to specially thank Professors Miguel Cazorla and Vicente Matellan (Editors-in-Chief of the Journal of Physical Agents) for the publication of this special issue and their support during the publishing process.

Acknowledgements

This special session was supported by CONSOLIDER-INGENIO 2010 under grant CSD2007-00022 and by the Spanish government and GVA funds under TIN2006-14630-C0301 and PROMETEO/2008/051 projects.

A Formal Model of Persuasion Dialogs for Interactions among Argumentative Software Agents

Andrés Muñoz and Juan A. Botía
Dpto. de Ingenería para la Información y las Comunicaciones
University of Murcia, Spain, C.P. 30100
E-mail: {amunoz,juanbot}@um.es

***Abstract*—Multi-agent systems (MAS) offer an approach to solve complex problems where data and control are inherently distributed among several agents. Normally, agents have to cope with conflicts based on inconsistent knowledge which arise when they exchange information. A relative common alternative to solve these conflicts is Argumentation. This technique allows agents to develop a coordination process based on the exchange of justifications supporting a piece of knowledge, with the aim of proving its validness. However, most argumentation frameworks are theoretical approaches to this problem. We have developed an Argumentation System Based on Ontologies (ASBO). It follows an engineering-oriented approach to materialize a software architecture which allows working with argumentation in MAS. Moreover, ASBO has also a formal model in the background. This paper introduces such formal model, in order to identify and unambiguously define the core elements that argumentation systems should include.**

***Index Terms*—Multi-agent systems engineering, Knowledge management, Argumentation, Persuasion dialogs.**

I. Introduction

AGENT-oriented paradigm [1] has been recognized as a powerful technique for modeling scenarios where a sophisticated software program autonomously acts on behalf of the users in order to fulfill their demands. As the complexity of the problems modeled in this paradigm grows, it is required that multiple agents can work together [2]–[4]. As a result, multi-agent system (MAS) engineering has been developed as a means of coordinating agents [5]. One particularity in MAS approaches is that no agent holds a complete vision of the problem faced (i.e., data and control are distributed). In these systems, the beliefs of one agent compound the personal view that such an agent has with respect to the part of the problem it is in charge of. Due to the distributed nature of multi-agent systems, these beliefs may be incomplete and overlapped with other agents' beliefs.

In this context, knowledge conflicts [6], [7] may arise from the situation where several agents have their own local knowledge base with incomplete and different information about the same domain. Moreover, each agent is responsible for maintaining the coherence of its knowledge base. Then, whenever agents exchange information, inconsistencies may arise in their local knowledge bases due to these different and/or incomplete agents' views of the state of affairs (e.g., a network administration scenario where some agents may believe that the access to a resource is granted, whereas others assert it as forbidden). As a result, agents must cope with these inconsistences as conflicts based on knowledge.

Classifications of knowledge conflicts usually divide them into two general types [8]: syntactic conflicts, so-called *contradictions*, and semantic ones. Contradictions may appear regardless how the domain is modeled (i.e., a fact and its negation, such as the network administration example above). Contrarily, semantic conflicts stick to the considered domain (e.g., classifying an object as square and rectangular in a domain where objects are restricted to be classified by a unique shape).

Although much of the work on solving knowledge conflicts in MAS has primarily been focused on syntactic conflicts [9]–[11], current proposals are also taken semantic conflicts into consideration [12], [13]. Note that this type of conflicts is more difficult to detect than syntactic ones, since they must be analyzed with respect to the meanings of a specific domain. We have proposed an approach for enabling agents to detect and solve both types of conflicts indistinctly. This approach consists in the combination of a formal knowledge model represented through ontologies and a mechanism for managing conflicts based on argumentation techniques. As a result, an argumentation system based on ontologies (ASBO henceforth) has been developed [14]. The elements of ASBO are briefly introduced here.

Firstly, the universe of discourse (i.e., the domain knowledge) is represented by means of a formal model based on Semantic Web [15] technologies. More precisely, we refer to OWL (Web Ontology Language) ontologies [16]. In this manner, the agents' beliefs are expressed as instances of these ontologies. Due to the addition of meta-information on this model, agents can support reasoning operations by means of inference engines (e.g., Pellet [17] or Jena [18]). Examples of reasoning operations are deductive processes to entail new knowledge, or consistency checking of ontologies to detect inconsistencies. Thus, detection of both semantic and syntactic knowledge conflicts are supported by these operations in a straightforward manner.

A common approach to coordinate agents to autonomously solve conflicts is by reaching agreements about the status of those conflicts. To this end, the alternative followed in ASBO resides in employing a *persuasion* dialog [19], [20]. Such kind of dialog consists in an exchange of opinions among agents that are for/against a piece of conflictive knowledge, with the aim of clarifying which opinion is the most acceptable.

Argumentation [21]–[23] is considered as a promising materialization of persuasion dialogs in multi-agent systems. In this manner, a negotiation protocol is defined via argumentation [24], which leads to a persuasion dialog where an agent tries to convince others about a specific proposal. But in this case, not only are the proposals exchanged. Furthermore, *arguments* (i.e. premises and rules used to derive those proposals) are also communicated. It allows agents to resolve conflicts more efficiently than just exchanging proposals, as proved elsewhere [25].

Different attacks (i.e., conflictive points of view on proposals, premises and rules) can be defined over arguments. These attacks could in turn be supported by other arguments, which may also be attacked. Hence, an argumentation process takes place by starting a persuasion dialog where agents exchange their arguments. Eventually, an acceptability status is determined for each argument depending on the development of the persuasion dialog. As a result, if an argument supporting a conflicting proposal is *accepted* by the agents involved in the dialog (i.e., the argument has no attacks or all its possible attacks have been invalidated by other arguments), such a proposal is also accepted. Contrarily, a *defeated* argument (i.e., an argument attacked by an accepted one) means that its proposal is not accepted.

While details on the above elements can be found elsewhere [14], [26], this paper is devoted to formalize the persuasion dialog framework used in ASBO for exchanging arguments. In this manner, we identify and formally define the elements which should be present in persuasion dialogs developed within an argumentation process. Section II presents the formal model of such a framework. Section III illustrates how this framework is used through an example. Section IV outlines most important conclusions and future works.

II. Formalization of the persuasion dialog framework in ASBO

The ASBO approach is restricted to the world of software agents. Its main goal is to provide them with an effective mechanism to solve knowledge conflicts by exchanging arguments through a persuasion dialog. ASBO agents are structured in a layered manner, from an architectural point of view. Figure 1 shows a representation of a couple of ASBO agents based on a block diagram. The top layer is related to argumentation tasks. It includes the definitions of the argumentation system (i.e., definitions of arguments and relationships of attack and defeat between them) and the persuasion dialog framework. The middle layer constitutes the formal description model of the universe of discourse. It contains a common representation (i.e., OWL ontologies) to express the agents' beliefs. Thus, this layer includes the domain knowledge upon which arguments are built. The bottom layer encapsulates a particular implementation for a specific agent platform.

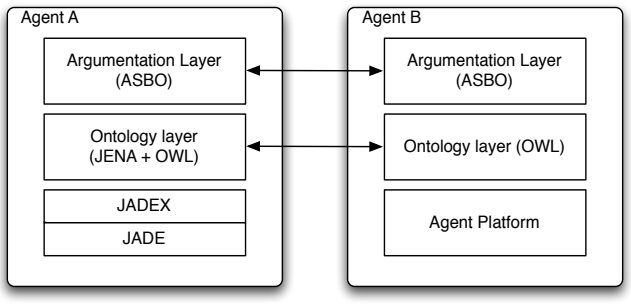

Fig. 1. Two ASBO agents showing the layered disposition of elements.

Details about the definition of the argumentation system are given in [14], whereas the middle and bottom layers are explained in [26]. The rest of the section is devoted to the persuasion dialog framework in the Argumentation layer, by giving a formal description of the communication language, the interaction protocol, the context and state of the dialog, the effect rules and the termination and outcome conditions. The selection of these elements is based on the Prakken's framework [20] and they are formalized here to define the persuasion dialogs hold in ASBO.

A. The communication language

The first element that needs to be defined in the ASBO persuasion dialog framework is a communication language for sending and receiving beliefs and the arguments supporting them. In this paper, the ASBO communication language is defined as the set of messages which can be exchanged among agents to this end. It will be denoted by $ASBO_{cm}$.

Table I summarizes all the messages defined to enable persuasion dialogs in ASBO. They are defined with (1) the form of the utterance (i.e., the performative and message body), (2) the semantic of the utterance (i.e., the *locutionary* force), (3) the intention of the emitter (i.e., the *illocutionary* act) and (4) the effect on the receiver (i.e., the *perlocutionary* effect).

Each performative ($Claim$, Why, $Since$, ...) defines the type of communicative act of the $ASBO_{cm}$ messages, explained below. On the other hand, the message body is formed by either assertions (i.e., agents' beliefs) and arguments. Assertions are denoted by φ, whereas arguments are denoted by S. An argument is defined as

$$S = (\varphi, \Phi),$$

where φ is the assertion representing the conclusion of the argument, and Φ is the support set which justifies that conclusion. In turn, Φ is composed of assertions called *premises*, denoted here by σ. The language for expressing both assertions and arguments is given by the Ontology layer in figure 1, namely OWL.

Let us see now the intended communicative act of each message. $[Claim\ \varphi]$ allows an agent to communicate the assertion φ to other agents, with the aim of convincing them about its validness. $[Why\ \varphi]$ enables an agent to ask for arguments about φ. Each argument is communicated through $[\varphi\ Since\ S]$. The message $[Concede\ \varphi]$ is used to inform that φ is accepted by the emitter. On the other hand, $[Retract\ \varphi]$ allows an agent to withdraw from supporting φ when such an agent does not have any argument for it or its arguments

TABLE I
AVAILABLE MESSAGES IN $ASBO_{cm}$

(1) Utterance	(2) Literal meaning	(3) Intention of the emitter	(4) Effect in the receiver
Claim φ	Statement of assertion φ	To impose φ to other agents	The speaker is associated to φ
Why φ	Challenge of assertion φ, seeking for arguments supporting it	To obtain arguments for φ	The assertion φ must be justified with arguments
φ **Since** S ($S = (\varphi, \Phi)$)	Disclosure of an argument S which supports φ	To prove φ as a justified assertion	The speaker is associated to premises σ_i in $\Phi, i = 1..n$
Concede φ	Assumption of assertion φ	To announce that the speaker agrees to φ	The speaker is associated to φ
Retract φ	Rejection of assertion φ	To withdraw from φ because no valid argument can be found for it	The speaker withdraws from the assertion φ
Accept S ($S = (\varphi, \Phi)$)	Acceptation of argument S	To update the speaker's knowledge according to the argument S	The speaker accepts the conclusion φ supported by S
No-Response m	Withdrawal from answering a message m	To announce that the speaker has no valid responses to m	The speaker can not respond a message m

are not valid anymore (i.e., they have been defeated by other arguments during the persuasion dialog). The message $[Accept\ S]$ is used when the emitter agrees to the argument S. Finally, the message $[No\text{-}Response\ m]$ is useful when an agent can not answer a message m and the dialog is not finished yet (see Section II-F). Observe that the performatives in Table I are not FIPA standard [27], since ASBO agents do not necessarily follow it. As a result, we have extended the FIPA performative set to include them. This subject requires a deeper analysis and it is beyond the scope of this paper.

B. The interaction protocol

After defining the messages in $ASBO_{cm}$, it is necessary a protocol for controlling the exchange of such messages between agents. The goal of this protocol is to delimit how to start, follow and end a persuasion dialog in ASBO. These dialogs are held between two agents, namely a *proponent* and an *opponent*. The proponent puts forward its arguments to persuade the opponent to accept an initial assertion. On the other hand, the opponent tries to prove that such an assertion is not valid through its own arguments, if possible (i.e., the opponent has beliefs which are inconsistent with the proponent's assertions).

The interaction protocol is defined in ASBO by using an AUML diagram [28] (see figure 2). The symbol $\diamond$ represents a decision node, in which only one message is selected. Let us denote this protocol with P_{ASBO}. P_{ASBO} starts when the proponent *claims* an assertion φ. The opponent may respond *conceding* that assertion if it agrees to such a claim, or *asking for* an argument supporting φ through the performative *Why*. In the latter case, the proponent *retracts* φ if no arguments can be built for it or all them are defeated. If the proponent has a valid argument S supporting φ, it is sent through the performative *Since*.

When receiving an argument S, the opponent has three options: (1) To *accept* S if the opponent agrees to it; (2) to *ask* for arguments supporting a premise σ_i in the support set of S; or (3) to attack the conclusion or premises in S by giving another argument T. The concept of *attack* to an assertion (i.e., conflict between pieces of knowledge) is represented by the negation symbol $\neg$ in figure 2. When the opponent answers as in the case (2), the proponent must now give an argument for the inquired premise or retract it. In the case (3), the proponent has now the same three options listed above with respect to the argument T (i.e., to accept T, to challenge a premise in T, or to attack T). The rest of the protocol can easily be followed from figure 2. Moreover, the message $[No\text{-}Response\ m]$ could be used as response for the rest of messages in $ASBO_{cm}$, although it has not been included in the figure for simplicity. This message has no responses and it will only be used when an agent can not attack the message body of m neither it can be accepted.

Fig. 2. AUML diagram of the ASBO persuasion protocol.

Observe that the protocol in figure 2 offers several options to answer a message. These options are called the *answer result set* (ARS) of a message. Thus, P_{ASBO} could be expressed as a function which takes a message and returns the correspondent ARS for that message according to the protocol in figure 2. In the rest of the paper, we will denote its use as $P_{ASBO}(m)$, where m is a message. For example, suppose that an agent receives the message $[Claim\ \varphi]$[1]. By applying P_{ASBO} to it, the agent obtains the following answer result set

$$P_{ASBO}([Claim\ \varphi]) = \begin{array}{l}\{[Why\ \varphi], [Concede\ \varphi], \\ [No-Response(Claim\ \varphi)]\}\end{array}$$

Notice that the ARS for a message is not sufficient to generate a suitable response. Therefore, an additional piece of information is needed for that: the context of the dialog.

C. The context and the state of a dialog

The context of a dialog is the set of information that enables an agent to take a decision about which message in the ARS is suitable to be used as an answer. Each agent maintains its own context, which is updated as the dialog progresses (see Section II-D). Hence, the context of a dialog for one agent in a moment t includes the sequence of messages emitted by the agent until that moment, the beliefs that such an agent holds, and the assertions emitted by the other agent. Formally, a context in ASBO is defined as follows:

Definition 1 (Context): Let us consider P, O as two agents which are using P_{ASBO} to exchange messages. Let Δ_P^t be the set of beliefs that P holds in a particular moment t. Let $H_{P,O}^t$ be the set of messages emitted by P to O in t, and let $\Pi_{P,O}^t$ be the set of assertions that O has communicated to P in t. Then, a context for the agent P in a moment t with respect to a dialog with the agent O is defined as the 3-tuple (Δ_P^t, $H_{P,O}^t$, $\Pi_{P,O}^t$), denoted as $C_{P,O}^t$. Analogously, the context for the agent O is $C_{O,P}^t$ = (Δ_O^t, $H_{O,P}^t$, $\Pi_{O,P}^t$).

Notice that $H_{P,O}^t$ contains the history of all messages sent by P to O until t. This history is used together with P_{ASBO} as follows: when the agent P needs to know which utterances may be answered, it uses P_{ASBO} to obtain the corresponding ARS. After this, only those messages compound by a performative and a message body which do not cause cycles can be emitted. Precisely, messages m causing cycles are those which for a particular message $m' \in H_{P,O}^t$, the performative and message body are the same in m and m' (i.e., either assertions φ in m and φ' in m' or arguments S in m and S' in m' are equivalent). Then, the remaining answers in the ARS are evaluated using the agent's beliefs, Δ_P^t, and the assertions the other agent is committed to, $\Pi_{P,O}^t$, so as to decide which answer is the optimal. Details about this evaluation are beyond the scope of this paper, and we suppose it is available in the form of the function 1 given below.

Once the context of a dialog for an agent has been given, it is possible to define a strategy function which, starting from the ARS of a message, it generates a single response:

Function 1 (Strategy function $\mathcal{P}$): We define the strategy function $\mathcal{P}$ in ASBO as

$$\mathcal{P} : ARS \times C \longrightarrow ASBO_{cm},$$

where ARS is the set of all possible answer result sets and C is the set of all possible contexts. Then, for two particular agents P, O involved in an exchange of messages in an instant of time t, $\mathcal{P}$ is such that $\forall ars \in ARS$, $\mathcal{P}(ars, C_{P,O}^t) \notin H_{P,O}^t$. Analogously, $\mathcal{P}(ars, C_{O,P}^t) \notin H_{O,P}^t$.

Apart from the context, agents also need to maintain the *state* of the dialog. This state contains the set of all messages exchanged between agents until the moment t, which agent has the turn of speaking in t, and which message is currently being processed. In ASBO, it is defined as follows:

Definition 2 (State): Let P, O be two agents using P_{ASBO} to exchange messages in a moment t. Let $\mathcal{T} \in \{P, O\}$ be the agent responsible for sending a message in $t+1$, and let m_t be the current message being processed in t by $\mathcal{T}$. Then, the state of that exchange of message in t is the 3-tuple $(\mathcal{T}, m_t, H_{P,O}^t \cup H_{O,P}^t)$, denoted as $\mathcal{S}^t$.

Observe that $\mathcal{S}^t$ is the same for both agents P and O. The state of a dialog is updated by means of the effects associated to the $ASBO_{cm}$ messages. Such effects are defined in the form of *effect rules*, explained in the next section.

D. Effect rules

$ASBO_{cm}$ messages are communicative acts. Therefore, the agents' contexts and the state $\mathcal{S}$ of a dialog must be updated according to effects of these acts. The update of $\mathcal{S}$ may be performed through two different directions. Suppose $\mathcal{S}^t = (P, m_t, H_{P,O}^t \cup H_{O,P}^t)$. Then, some messages will update $\mathcal{S}$ in $t+1$ as

$$\mathcal{S}^{t+1} = (O, m_{t+1}, H_{P,O}^{t+1} \cup H_{O,P}^{t+1}),$$

i.e. the agent P emitting a message in $\mathcal{S}^t$ is now the receiver of the next one in $\mathcal{S}^{t+1}$. Some other messages, namely *surrendered* (those using the performatives $Concede$, $Retract$, $Accept$ and $No\text{-}Response$), update $\mathcal{S}$ in $t+1$ as

$$\mathcal{S}^{t+1} = (P, m_{t'}, H_{P,O}^{t+1} \cup H_{O,P}^{t+1}),$$

where $t' < t$ and $m_{t'} \in H_{O,P}^t$, i.e. P must again utter a new message in $\mathcal{S}^{t+1}$ as response to a previous message sent by O. This situation occurs due to the emitter of a surrendered message has reached the end of an argumentation line in $\mathcal{S}^t$, and therefore it is still the responsible for continuing the exchange of messages by exploring another argumentation line (if possible). In order to define these exploration mechanisms, ASBO agents need a *backward* function:

Function 2 (Backward function $\mathcal{B}$): We define a backward function in ASBO as

$$\mathcal{B} : ASBO_{cm} \times S \longrightarrow ASBO_{cm},$$

[1] And the dialog is in the appropriate state for considering such message as correct.

where S is the set of all possible states. Then,

$$\mathcal{B}(m_{t+n}, S^{t+n}) = m_t$$

for a particular message m_{t+n} and the state of the exchange of messages $\mathcal{S}^{t+n} = (E, m_{t+n}, H^{t+n}_{E,R} \cup H^{t+n}_{R,E})$ between two agents E and R in $t+n$, $n > 1$, if the following conditions hold:

1) Let m_t, m_{t+1} and m_{t+n} be messages exchanged among E and R until the instant $t+n$, where $m_t, m_{t+n} \in H^{t+n}_{R,E}$ and $m_{t+1} \in H^{t+n}_{E,R}$, and
2) $m_{t+1} = \mathcal{P}(P_{ASBO}(m_t), C^t_{E,R})$, and
3) $m_{t+n} = \mathcal{P}(P_{ASBO}(m_{t+1}), C^{t+n-1}_{R,E})$.

Then, if E emits a surrendered message in $\mathcal{S}^{t+n}$ as an answer to m_{t+n}, it still has the turn of speaking. Now, to discover which previous message emitted by R must be responded, E applies $\mathcal{B}(m_{t+n}, \mathcal{S}^{t+n})$, and then

$$\mathcal{S}^{t+n+1} = (E, \mathcal{B}(m_{t+n}, \mathcal{S}^{t+n}), H^{t+n+1}_{E,R} \cup H^{t+n+1}_{R,E}).$$

A case of use of $\mathcal{B}$ is given in Section III. Let us see now the effect rules of $ASBO_{cm}$ messages.

Definition 3 (Effect rules): Let E, R be the respective emitter and receiver agents of an $ASBO_{cm}$ message m_{t+1}. Let $\mathcal{S}^t = (E, m_t, H^t_{E,R} \cup H^t_{R,E})$ be the state of the exchange of messages between these agents in the instant t. Let $C^t_{E,R}$ and $C^t_{R,E}$ be the contexts of agents E and R in t, respectively, such as $m_{t+1} = \mathcal{P}(P_{ASBO}(m_t), C^t_{E,R})$. Then, the set of effect rules for m_{t+1} is defined as follows:

ER1. If $m_{t+1} = [Claim\ \varphi]$, then

- $C^{t+1}_{E,R} = (\Delta^t_E, H^t_{E,R} \cup \{m_{t+1}\}, \Pi^t_{E,R})$,
- $C^{t+1}_{R,E} = (\Delta^t_R, H^t_{R,E}, \Pi^t_{R,E} \cup \{\varphi\})$, and
- $\mathcal{S}^{t+1} = (R, m_{t+1}, H^t_{E,R} \cup H^t_{R,E} \cup \{m_{t+1}\})$.

ER2. If $m_{t+1} = [Why\ \varphi]$, then

- $C^{t+1}_{E,R} = (\Delta^t_E, H^t_{E,R} \cup \{m_{t+1}\}, \Pi^t_{E,R})$,
- $C^{t+1}_{R,E} = C^t_{R,E}$, and
- $\mathcal{S}^{t+1} = (R, m_{t+1}, H^t_{E,R} \cup H^t_{R,E} \cup \{m_{t+1}\})$.

ER3. If $m_{t+1} = [\varphi\ Since\ S], S = \{\varphi, \Phi\}$, then

- $C^{t+1}_{E,R} = (\Delta^t_E, H^t_{E,R} \cup \{m_{t+1}\}, \Pi^t_{E,R})$,
- $C^{t+1}_{R,E} = (\Delta^t_R, H^t_{R,E}, \Pi^t_{R,E} \cup \Phi)$, and
- $\mathcal{S}^{t+1} = (R, m_{t+1}, H^t_{E,R} \cup H^t_{R,E} \cup \{m_{t+1}\})$.

ER4. If $m_{t+1} = [Concede\ \varphi]$, then

- $C^{t+1}_{E,R} = (\Delta^t_E, H^t_{E,R} \cup \{m_{t+1}\}, \Pi^t_{E,R})$,
- $C^{t+1}_{R,E} = (\Delta^t_R, H^t_{R,E}, \Pi^t_{R,E} \cup \{\varphi\})$, and
- $\mathcal{S}^{t+1} = (E, \mathcal{B}(m_t, \mathcal{S}^t), H^t_{E,R} \cup H^t_{R,E} \cup \{m_{t+1}\})$.

ER5. If $m_{t+1} = [Retract\ \varphi]$, then

- $C^{t+1}_{E,R} = (\Delta^t_E - \{\varphi\}, H^t_{E,R} \cup \{m_{t+1}\}, \Pi^t_{E,R})$,
- $C^{t+1}_{R,E} = (\Delta^t_R, H^t_{R,E}, \Pi^t_{R,E} - \{\varphi\})$, and
- $\mathcal{S}^{t+1} = (E, \mathcal{B}(m_t, \mathcal{S}^t), H^t_{E,R} \cup H^t_{R,E} \cup \{m_{t+1}\})$.

ER6. If $m_{t+1} = [Accept\ S], S = \{\varphi, \Phi\}$, then

- $C^{t+1}_{E,R} = (\Delta^t_E \cup \{\varphi\}, H^t_{E,R} \cup \{m_{t+1}\}, \Pi^t_{E,R})$,
- $C^{t+1}_{R,E} = (\Delta^t_R, H^t_{R,E}, \Pi^t_{R,E} \cup \{\varphi\})$, and
- $\mathcal{S}^{t+1} = (E, \mathcal{B}(m_t, \mathcal{S}^t), H^t_{E,R} \cup H^t_{R,E} \cup \{m_{t+1}\})$.

ER7. If $m_{t+1} = [No\text{-}Response\ m_t]$, then

- $C^{t+1}_{E,R} = (\Delta^t_E, H^t_{E,R} \cup \{m_{t+1}\}, \Pi^t_{E,R})$,
- $C^{t+1}_{R,E} = C^t_{R,E}$, and
- $\mathcal{S}^{t+1} = (E, \mathcal{B}(m_t, \mathcal{S}^t),\ H^t_{E,R} \cup H^t_{R,E} \cup \{m_{t+1}\})$.

Observe that when E utters m_{t+1} as a surrendered message ($Concede$, $Retract$, $Accept$ and $No\text{-}Response$), it still keeps the turn of speaking in $\mathcal{S}^{t+1}$, and the current message to be processed in that instant is $\mathcal{B}(m_t, \mathcal{S}^t)$. In other words, when E accepts an R's premise or argument, or it withdraws from an assertion previously claimed by itself, then E must give a new answer to a previous message sent by R (if possible).

E. ASBO persuasion dialog

After defining the previous elements (i.e., communication language, interaction protocol, agents' contexts, state of an exchange of messages and effect rules), persuasion dialogs can now be built as a sequence of messages according to such definitions. Thus, the formal definition of an ASBO persuasion dialog follows.

Definition 4 (ASBO persuasion dialog): Let agents P, O be the proponent and opponent of an assertion φ, respectively. Then, a sequence $d = [m_1, m_2, \ldots, m_n]$ is an ASBO persuasion dialog about φ if and only if:

1) $m_1 = [Claim\ \varphi]$, such as $m_1 \in H^n_{P,O}$, and
2) $\forall m_i \in d, i > 1$,

$$m_i = \mathcal{P}(P_{ASBO}(m_{i'}), C^{i-1}_{\mathcal{T},X}),$$

such as $1 \leq i' < i$, and

$$\mathcal{S}^{i-1} = (\mathcal{T}, m_{i'}, H^{i-1}_{P,O} \cup H^{i-1}_{O,P}),$$

where $X \equiv O$ if $\mathcal{T} \equiv P$ and $X \equiv P$ if $\mathcal{T} \equiv O$, and

$$m_{i'} = \mathcal{B}(m_{i-2}, \mathcal{S}^{i-2})$$

iff m_{i-1} is a surrendered message, or $m_{i'} = m_{i-1}$ otherwise, and

3) $\forall m_i, m_{i'} \in d$, if $i \neq i'$, $m_i \not\equiv m_{i'}$.

The first condition states that the dialog is always commenced by P claiming the assertion φ. The second condition defines each message m_i in the dialog (except the first one) as a result of applying $\mathcal{P}$ to a previous message $m_{i'}$ with respect to the context $C^{i-1}_{\mathcal{T},X}$ of the agent $\mathcal{T}$ holding the turn of speaking. Finally, the last condition avoids cycles in the dialog

since the repetition of message is not allowed. Although the function $\mathcal{P}$ forbids an agent to repeat an utterance, this last condition is necessary to prevent an agent from emitting the same message previously sent by the other agent.

F. Termination and outcome conditions

There are three circumstances in the ASBO persuasion framework that produce the end of a dialog discussing the assertion φ. In the first one, the opponent emits $[Concede\ \varphi]$ if it agrees to that claim or after accepting an argument supporting φ, since it can not find any valid opposition to such an argument. In the second one, the proponent utters $[Retract\ \varphi]$ when all the arguments supporting φ have been invalidated (i.e., defeated). Finally, the third condition occurs when the proponent emits $[No\text{-}Response\ [Why\ \varphi]]$ or the opponent utters $[No\text{-}Response\ [Claim\ \varphi]]$. This last condition means that one of the agents withdraws from the dialog without accepting or rejecting the initial claim φ. This situation may occur when some arguments for and against φ are incomparable, and neither are they accepted nor defeated.

Moreover, following these termination conditions, the outcome of the dialog can also be defined. If the first condition holds, the proponent wins and the opponent accepts φ. If the second condition holds, then the opponent wins and the proponent must reject φ. In the third case, none of the agents wins and φ is undecided with respect to them. The termination and outcome conditions are formally expressed next.

Definition 5 (Termination and outcome conditions): Let agents P, O be the proponent and opponent of an assertion φ, respectively. Let d be an ASBO persuasion dialog between P and O about φ, where $m_1 \in d$ is such that $m_1 = [Claim\ \varphi]$. Moreover, let S^t be the state of d in the instant t. Then, the dialog d terminates when:

1) $\mathcal{S}^t = (O, m_t, H^t_{P,O} \cup H^t_{O,P})$, and $m_{t+1} = [Concede\ \varphi]$, or
2) $\mathcal{S}^t = (P, m_t, H^t_{P,O} \cup H^t_{O,P})$, and $m_{t+1} = [Retract\ \varphi]$, or
3) $\mathcal{S}^t = (\mathcal{T}, m_t, H^t_{P,O} \cup H^t_{O,P})$, and
   - $\mathcal{T} \equiv P$ and $m_{t+1} = [No\text{-}Response\ [Why\ \varphi]]$, or
   - $\mathcal{T} \equiv O$ and $m_{t+1} = [No\text{-}Response\ [Claim\ \varphi]]$.

In the first case, P wins the dialog and φ is accepted. In the second case, O is the winner and φ is not accepted. Otherwise, φ is undecided.

III. An example of persuasion dialog in ASBO

In order to illustrate all the ideas explained so far, an example of a complete ASBO persuasion dialog is given next. Let us consider an agent A_{Ex} supporting an assertion φ by means of two different arguments, S=(φ, Φ_S) and R=(φ, Φ_R). Now, let B_{Ex} be an agent with an argument $T = (\psi, \Phi_T)$, where ψ defeats a premise in Φ_S (i.e., T undercuts S). Moreover, B_{Ex} does not have any opinion related to φ. A_{Ex} tries to convince B_{Ex} about the validness of φ. Therefore, both agents start an ASBO persuasion dialog, being A_{Ex} the proponent by claiming φ, and B_{Ex} the opponent by requiring a valid support to that claim while trying to invalidate it. At the initial moment, the local knowledge bases of each agents are

$$\Delta^0_{A_{Ex}} = \{\{\varphi\} \cup \Phi_S \cup \Phi_R\},$$
$$\Delta^0_{B_{Ex}} = \{\{\psi\} \cup \Phi_T\}.$$

Then, the dialog is developed as shown next:

- At t=0,
  - $\mathcal{S}^0$=$(A_{Ex}, -, \emptyset)$,
  - $C^0_{A_{Ex}}$=$(\Delta^0_{A_{Ex}}, \emptyset, \emptyset)$, and
  - $C^0_{B_{Ex}}$=$(\Delta^0_{B_{Ex}}, \emptyset, \emptyset)$.

  Agent A_{Ex} seeks to persuade B_{Ex} to accept φ. Then, m_1=$[Claim\ \varphi]$.

- At t=1, m_1=$[Claim\ \varphi]$, and
  - $\mathcal{S}^1$=$(B_{Ex}, m_1, \{m_1\})$,
  - $C^1_{A_{Ex}}$=$(\Delta^0_{A_{Ex}}, \{m_1\}, \emptyset)$, and
  - $C^1_{B_{Ex}}$=$(\Delta^0_{B_{Ex}}, \emptyset, \{\varphi\})$.

  Agent B_{Ex} does not have any initial information to agree to φ, so it asks for an argument supporting it. Then, $\mathcal{P}(P_{ASBO}(m_1), C^1_{B_{Ex}})$ = m_2 = $[Why\ \varphi]$.

- At t=2, m_2 = $[Why\ \varphi]$, and
  - $\mathcal{S}^2$=$(A_{Ex}, m_2, \{m_1, m_2\})$,
  - $C^2_{A_{Ex}}$=$C^1_{A_{Ex}}$, and
  - $C^2_{B_{Ex}}$=$(\Delta^1_{B_{Ex}}, \{m_2\}, \{\varphi\})$.

  Agent A_{Ex} must communicate an argument justifying φ. As stated above, it owns arguments S and R for this. For the purpose of this example, suppose that argument S is the most appropriate alternative according to A_{Ex}'s strategy function. Then, $\mathcal{P}(P_{ASBO}(m_2), C^2_{A_{Ex}})$ = m_3 = $[\varphi\ Since\ S]$.

- At t=3, m_3 = $[\varphi\ Since\ S]$, and
  - $\mathcal{S}^3$ = $(B_{Ex}, m_3, \{m_1, m_2, m_3\})$,
  - $C^3_{A_{Ex}}$ = $(\Delta^2_{A_{Ex}}, \{m_1, m_3\}, \emptyset)$, and
  - $C^3_{B_{Ex}}$ = $(\Delta^2_{B_{Ex}}, \{m_2\}, \{\varphi \cup \Phi_S\})$.

  Agent B_{Ex} has the belief ψ which is in conflict with a premise in argument S. Moreover, ψ is supported by the argument T. Then, $\mathcal{P}(P_{ASBO}(m_3), C^3_{B_{Ex}})$ = m_4 = $[\psi\ Since\ T]$. Note that argument T defeats argument S since ψ undercuts a premise in Φ_S.

- At t=4, m_4 = $[\psi\ Since\ T]$, and
  - $\mathcal{S}^4$ = $(A_{Ex}, m_4, \{m_1, m_2, m_3, m_4\})$,
  - $C^4_{A_{Ex}}$ = $(\Delta^3_{A_{Ex}}, \{m_1, m_3\}, \{\psi \cup \Phi_T\})$, and
  - $C^4_{B_{Ex}}$ = $(\Delta^3_{B_{Ex}}, \{m_2, m_4\}, \{\varphi \cup \Phi_S\})$.

Fig. 3. Tree generated from the persuasion dialog between A_{Ex} and B_{Ex}.

Agent A_{Ex} does not have any attacks against argument T. Then, $\mathcal{P}(P_{ASBO}(m_4), C^4_{A_{Ex}}) = m_5 = [Accept\ T]$. Notice that m_5 is a surrendered message answering to m_4. In this manner, the argumentation line supporting φ through S is ended since such an argument has been defeated. Now, by applying $\mathcal{B}(m_4, \mathcal{S}^4)$, the message to process in $\mathcal{S}^5$ by A_{Ex} is m_2 again. This result is due to a new argumentation line can be opened from answering $m_2 = [Why\ \varphi]$ with another argument supporting φ.

- At t=5, $m_5 = [Accept\ T]$, and
  - $\mathcal{S}^5 = (A_{Ex}, m_2, \{m_1, m_2, m_3, m_4, m_5\})$,
  - $C^5_{A_{Ex}} = (\Delta^4_{A_{Ex}} \cup \{\psi\}, \{m_1, m_3, m_5\}, \{\psi \cup \Phi_T\})$, and
  - $C^5_{B_{Ex}} = (\Delta^4_{B_{Ex}}, \{m_2, m_4\}, \{\varphi \cup \Phi_S \cup \psi\})$.

Agent A_{Ex} still has the argument R supporting φ. This argument has not previously been uttered neither defeated. Then, $\mathcal{P}(P_{ASBO}(m_2), C^5_{A_{Ex}}) = m_6 = [\varphi\ Since\ R]$. Hence, it gives a new argument supporting the assertion φ, by answering to the message $[Why\ \varphi]$ again.

- At t=6, $m_6 = [\varphi\ Since\ R]$, and
  - $\mathcal{S}^6 = (B_{Ex}, m_6, \{m_1, m_2, m_3, m_4, m_5, m_6\})$,
  - $C^6_{A_{Ex}} = (\Delta^5_{A_{Ex}}, \{m_1, m_3, m_5, m_6\}, \{\psi \cup \Phi_T\})$, and
  - $C^6_{B_{Ex}} = (\Delta^5_{B_{Ex}}, \{m_2, m_4\}, \{\varphi \cup \Phi_S \cup \psi \cup \Phi_R\})$.

Agent B_{Ex} does not have any attacks against argument R. Then, $\mathcal{P}(P_{ASBO}(m_6), C^6_{B_{Ex}}) = m_7 = [Accept\ R]$. Observe that B_{Ex} emits a surrendered message in m_7 as a response to message m_6. In turn, m_6 is an answer to m_2. Finally, m_2 is answering to m_1. As a result, $\mathcal{B}(m_6, \mathcal{S}^6) = m_1$ is the message to be processed by B_{Ex} in $\mathcal{S}^7$.

- At t=7, $m_7 = [Accept\ R]$, and
  - $\mathcal{S}^7 = (B_{Ex}, m_1, \{m_1, m_2, m_3, m_4, m_5, m_6, m_7\})$,
  - $C^7_{A_{Ex}} = (\Delta^6_{A_{Ex}}, \{m_1, m_3, m_5, m_6\}, \{\psi \cup \Phi_T\ \cup \varphi)$, and
  - $C^7_{B_{Ex}} = (\Delta^6_{B_{Ex}} \cup \varphi, \{m_2, m_4, m_7\}, \{\varphi, \Phi_S, \psi, \Phi_R\})$.

Agent B_{Ex} has previously accepted the argument R supporting φ. As a result, it now agrees to such an assertion. Then, $\mathcal{P}(P_{ASBO}(m_1), C^7_{B_{Ex}}) = m_8 = [Concede\ \varphi]$.

Eventually, the dialog has finished since B_{Ex} has conceded the initial proposal φ. The dialog sequence is $[\lfloor Claim\ \varphi \rfloor, \lceil Why\ \varphi \rceil, \lfloor \varphi\ Since\ S \rfloor, \lceil \psi\ Since\ T \rceil, \lfloor Accept\ T \rfloor, \lfloor \varphi\ Since\ R \rfloor, \lceil Accept\ R \rceil, \lceil Concede\ \varphi \rceil]$, where $\lfloor p \rfloor$ is a message uttered by A_{Ex} and $\lceil p \rceil$ is a message uttered by B_{Ex}. Consequently, A_{Ex} has won the dialog, and φ is finally accepted by B_{Ex}. Figure 3 shows the dialog using a tree-based representation.

IV. Conclusions

Argumentation has demonstrated to be a useful approach when resolving conflicts during the exchange of proposals in a MAS. This is possible due to agents can use their rationality to build and attack arguments that justify those proposals. However, most argumentative approaches in MAS are theoretical. We have developed an Argumentation System Based on Ontologies, ASBO, with an engineering oriented approach. In this manner, ASBO is offered as a software architecture ready to be used by agents performing argumentation in conventional MAS. This paper has presented the formal background for the persuasion dialog framework included in such architecture. This formal model can then be used to identify the necessary elements to be defined for implementing persuasion dialogs within argumentation processes.

One future step in ASBO is to adopt a BDI agency paradigm. In this manner, an agent could select the optimal argument during the persuasion dialog according to its intentions and desires.

ACKNOWLEDGMENTS

This work has been supported by the Fundación Séneca grant "Programa de Ayuda a los grupos de excelencia 04552/GERM/06 " and thanks to the Spanish Ministerio de Ciencia e Innovación (MICINN) under the grant AP2006-4154 in frames of the FPU Program, the TIN2005-08501-C03-01 Program and the CICYT TIN2008-06441-C02-02 project.

REFERENCES

[1] M. Luck and J. J. Gómez-Sanz, Eds., *Agent-Oriented Software Engineering IX, 9th International Workshop, AOSE 2008, Estoril, Portugal, May 12-13, 2008, Revised Selected Papers*, ser. Lecture Notes in Computer Science, vol. 5386. Springer, 2009.

[2] S. Ossowski, J.-L. P. de-la Cruz, J. Z. Hernández, J.-M. Maseda, A. Fernández, M.-V. Belmonte, A. García-Serrano, J. M. Serrano, R. León, and F. Carbone, "Towards a generic multiagent model for decision support: Two case studies," in *CAEPIA*, ser. Lecture Notes in Computer Science, R. Conejo, M. Urretavizcaya, and J.-L. P. de-la Cruz, Eds., vol. 3040. Springer, 2003, pp. 597–607.

[3] N. dos Santos, F. M. Varejáo, and O. de Lira Tavares, "Multi-agent systems and network management - A positive experience on Unix environments," in *IBERAMIA 2002: Proceedings of the 8th Ibero-American Conference on AI*. London, UK: Springer-Verlag, 2002, pp. 616–624.

[4] D. Sharma and F. Shadabi, "An intelligent multi-agent design in healthcare management system," in *KES-AMSTA*, ser. Lecture Notes in Computer Science, N. T. Nguyen, G. Jo, R. J. Howlett, and L. C. Jain, Eds., vol. 4953. Springer, 2008, pp. 674–682.

[5] R. Choren, A. F. Garcia, H. Giese, H. fung Leung, C. J. P. de Lucena, and A. B. Romanovsky, Eds., *Software Engineering for Multi-Agent Systems V, Research Issues and Practical Applications*, ser. Lecture Notes in Computer Science, vol. 4408. Springer, 2007.

[6] A. Muñoz, J. A. Botía, F. J. García, G. Martínez, and A. F. G. Skarmeta, "Solving conflicts in agent-based ubiquitous computing systems: A proposal based on argumentation," in *Agent-Based Ubiquitous Computing*, ser. Atlantis Ambient And Pervasive Intelligence, E. Mangina, J. Carbó, and J. M. Molina, Eds., vol. 1, no. 1. Atlantis Press, May 2009, pp. 1–12.

[7] C. Tessier, L. Chaudron, and H.-J. Müller, Eds., *Conflicting agents: conflict management in multi-agent systems*. Norwell, MA, USA: Kluwer Academic Publishers, 2001.

[8] N. T. Nguyen, *Advanced Methods for Inconsistent Knowledge Management (Advanced Information and Knowledge Processing)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[9] J. Fox, P. Krause, and S. Ambler, "Arguments, contradictions and practical reasoning," in *ECAI '92: Proceedings of the 10th European conference on Artificial intelligence*. New York, NY, USA: John Wiley & Sons, Inc., 1992, pp. 623–627.

[10] M. N. Huhns and D. M. Bridgeland, "Multiagent truth maintenance," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 21, no. 6, pp. 1437–1445, 1991.

[11] G. K. Kraetzschmar, *Distributed Reason Maintenance for Multiagent Systems*, J. Siekmann and J. G. Carbonell, Eds. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1997.

[12] S. Ram and J. Park, "Semantic conflict resolution ontology (SCROL): An ontology for detecting and resolving data and schema-level semantic conflicts," *Transactions on Knowledge and Data Engineering*, vol. 16, no. 2, pp. 189–202, 2004.

[13] K. Lee and M. Kim, "A novel approach for conflict resolution in context-awareness using semantic unification of multi-cognition," in *PRIMA '08: Proceedings of the 11th Pacific Rim International Conference on Multi-Agents*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 267–274.

[14] A. Muñoz and J. A. Botía, "ASBO: Argumentation system based on ontologies," in *Cooperative Information Agents XII*, ser. Lecture Notes in Artificial Intelligence, M. Klusch, M. Pechoucek, and A. Polleres, Eds., vol. 5180. Springer, 2008, pp. 191–205.

[15] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, vol. 284, no. 5, pp. 34–43, 2001.

[16] M. Dean, D. Connoll, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein, "Web Ontology Language (OWL)," W3C, Tech. Rep., 2004. [Online]. Available: http://www.w3.org/TR/owl-ref/

[17] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical OWL-DL reasoner," *Journal of Web Semantics*, vol. 5, no. 2, pp. 51–53, 2007.

[18] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson, "Jena: implementing the Semantic Web recommendations," in *Proceedings of the 13th international World Wide Web conference*. ACM Press, 2004, pp. 74–83.

[19] G. Brewka, "Dynamic argument systems: A formal model of argumentation processes based on situation calculus," *JLC: Journal of Logic and Computation*, vol. 11, 2001.

[20] H. Prakken, "Coherence and flexibility in dialogue games for argumentation," *Journal of Logic and Computation*, vol. 15, no. 6, pp. 1009–1040, 2005.

[21] D. V. Carbogim, D. Robertson, and J. Lee, "Argument-based applications to knowledge engineering," *Knowledge Engineering Review*, vol. 15, no. 2, pp. 119–149, 2000.

[22] N. Maudet, S. Parsons, and I. Rahwan, Eds., *Argumentation in Multi-Agent Systems, Third International Workshop, ArgMAS 2006, Hakodate, Japan, May 8, 2006, Revised Selected and Invited Papers*, ser. Lecture Notes in Computer Science, vol. 4766. Springer, 2007.

[23] I. Rahwan and B. Banihashemi, "Arguments in OWL: A Progress Report," in *Computational Models of Argument (COMMA)*, ser. Frontiers in Artificial Intelligence and Applications, P. Besnard, S. Doutre, and A. Hunter, Eds., vol. 172. IOS Press, 2008, pp. 297–310.

[24] S. Parsons, C. Sierra, and N. R. Jennings, "Agents that reason and negotiate by arguing," *Journal of Logic and Computation*, vol. 1998, no. 3, pp. 261–292, 1998.

[25] N. Karunatillake and N. R.Jennings, "Is it worth arguing?" in *Proc. of Argumentation in Multi-Agent Systems (ArgMAS)*, Springer-Verlag, Ed., vol. LNAI 3366, New York, USA, 2004, pp. 134–250.

[26] A. Muñoz, A. Sánchez, and J. A. Botía, "A software architecture for an argumentation-oriented multi-agent system," in *7th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS'09)*, ser. Advances in Intelligent and Soft Computing, Y. Demazeau, J. Pavn, J. Corchado, and J. Bajo, Eds., vol. 55. Springer, 2009, pp. 197–207.

[27] FIPA, "Fipa communicative act library specification," Tech. Rep., December 2002.

[28] B. Bauer, J. P. Müller, and J. Odell, "Agent UML: A formalism for specifying multiagent software systems," in *AOSE*, ser. Lecture Notes in Computer Science, P. Ciancarini and M. Wooldridge, Eds., vol. 1957. Springer, 2000, pp. 91–104.

A Hybrid Agent-based Classification Mechanism to Detect Denial of Service Attacks

Cristian I. Pinzón, Juan F. De Paz, Sara Rodríguez, Javier Bajo and Juan M. Corchado

***Abstract*—This paper presents the core component of a solution based on agent technology specifically adapted for the classification of SOAP messages. The messages can carry out attacks that target the applications providing Web Services. One of the most common attacks requiring novel solutions is the denial of service attack (DoS), caused for the modifications introduced in the XML of the SOAP messages. The specifications of existing security standards do not focus on this type of attack. This article presents an advanced mechanism of classification designed in two phases incorporated within a CBR-BDI Agent type. This mechanism classifies the incoming SOAP message and blocks the malicious SOAP messages. Its main feature involves the use of decision trees, fuzzy logic rules and neural networks for filtering attacks. These techniques provide a mechanism of classification with the self-adaption ability to the changes that occur in the patterns of attack. A prototype was developed and the results obtained are presented in this study.**

***Index Terms*—multi-agent systems, case-based reasoning, DoS Attack, SOAP message, XML security.**

I. Introduction

THE communication based on Service Oriented Architecture Web (SOA) is carried out by XML-based messages, called SOAP messages. This message exchange process is one of the key elements required in SOA environments for system integration [1]. The SOAP message payload often consists of sensitive information, which is sent through insecure channels such as HTTP connections. If a malicious user playing the role of a middleman intercepts a message between sender and recipient, it can result in a series of malicious tasks carried out over the captured message.

DoS attacks are due to the fact that XML messages must be parsed in the server, which opens the possibility of an attack if the messages themselves are not well structured or if they include some type of malicious code. Resources available in the server (memory and CPU cycles) of the provider can be drastically reduced or exhausted while a malicious SOAP message is being parsed. A DoS attack is successfully carried out when it manages to severely compromise legitimate user access to services and resources.

A number of technologies and solutions have been proposed for addressing the secure exchange of SOAP message such as WS-Security [2], WS-SecurityPolicy [3], WS-Trust [4], WS-SecureConversation [5] etc. All these standards focus on the aspects of message integrity and confidentiality and user authentication and authorization. None of the WS-Security Standards provide full security, leaving gaps that can be exploited by any malicious user.

Then, it is necessary to investigate in novel methods to protect the servers from denial of services attacks (DoS), which cause malicious or altered Web Services, and affect the availability of the Web Services [6]. This paper presents the core component of a strong solution based on a multi-agent architecture for tackling the security issue of the Web Service. This core is embedded in a CBR-BDI [7] deliberative agent based on the BDI (Belief, Desire, Intention) [8] model specifically adapted for preventing many attacks over web services. Our study applies a solution in two phases that include novel case-based reasoning (CBR) [9] classification mechanisms. The first phase incorporates decision tree and fuzzy logic rules [10] while the second phase incorporates neural networks capable of making short term predictions [11]. The idea of a CBR mechanism is to exploit the experience gained from similar problems in the past and to adapt a successful solution to the current problem. The CBR engine initiates what is known as the CBR cycle, which is comprised of 4 phases. The CBR-BDI agent explained in this work uses the CBR concept to gain autonomy and improve its problem-solving capabilities. The approach presented in this paper is entirely new and offers a different way to confront the security problem in SOA environments.

The rest of the paper is structured as follows: section 2 presents the problem that has prompted most of this research. Section 3 focuses on the structure of the classifiers agents which facilitates classification of SOAP message, and section 4 provides a detailed explanation of the classification model integrated within the type of classifier agent. Finally, section 5 presents the conclusions obtained by the research.

II. Web Service Security Problem Description

A web service is a software module designed to support interaction between heterogeneous groups within a network. In order to obtain interoperability between platforms, communication between web servers is carried out via an exchange of messages. These messages, referred to as SOAP messages, are based on standard XML (eXtensible Markup Language) and are primarily exchanged using HTTP (Hyper Text Transfer Protocol) [12].

One of the most frequent techniques of a DoS attack is to exhaust available resources (memory, CPU cycles, and

Cristian I. Pinzon is with The Technological University of Panama.
E-mail: cristian.pinzon@utp.ac.pa
Juan F. De Paz is with University of Salamanca.
E-mail: fcofds@usal.es
Sara Rodríguez is with University of Salamanca.
E-mail: srg@usal.es
Javier Bajo is with University of Salamanca.
E-mail: jbajope@usal.es
Juan M. Corchado is with University of Salamanca.
E-mail: corchado@usal.es

bandwidth) on the host server. The probability of a DoS attack increases with applications providing web services because of their intrinsic use of the XML standard. The server uses a parser, such as DOM, Xerces, etc. to syntactically analyze all incoming XML formatted SOAP messages. When the server draws too much of its available resources to parse SOAP messages that are either poorly written or include a malicious code, it risks becoming completely blocked. Attacks usually occur when the SOAP message either comes from a malicious user or is intercepted during its transmission by a malicious node that introduces different kinds of attacks.

Security is one of the greatest concerns within web service implementations. The following list contains descriptions of different types of attacks, compiled from those noted in [13], [14], [15].

- Oversize Payload: When it is executed, it reduces or eliminates the availability of a web service while the CPU, memory or bandwidth are being tied up by a massive message dispatch with a large payload.
- Coercive Parsing: Just like a message written with XML, an XML parser can analyze a complex format and lead to a denial of service attack because the memory and processing resources are being used up.
- Injection XML: This is based on the ability to modify the structure of an XML document when an unfiltered user entry goes directly to the XML stream or the message is captured and modified during its transmission.
- Parameter Tampering: A malicious user employs web service entries to manually or automatically (dictionaries attack) execute different types of tests and produce an unexpected response from the server.
- SOAP header attack: Some SOAP message headers are overwritten while they are passing through different nodes before arriving at their destination. It is possible to modify certain fields with malicious code.
- Replay Attack: Sent messages are completely valid, but they are sent en masse over a small time frame in order to overload the web service.

All web service security standards focus on strategies independent from DoS attacks [6]. In the following, we will revise those works that focus on denial of web service attacks and will compare to our approach.

A "XML Firewall"is proposed by [13]. The architecture of the XML Firewall is divided into three modules, namely Core Engine, Administrative Interface, and Database. The Core engine is the main component that processes and handles SOAP messages. Messages that are sent to a Web Service are intercepted and parsed to check the validity and the authenticity of the contents. If the contents of the messages do not conform to the policies that have been set, the messages will be dropped by the firewall. Three successfully implemented filtering policies, namely message size filtering, syntax parsing, and XML schema validation have been tested with valid and invalid SOAP messages. Gruschka and Luttenberger [6] propose an application level gateway system "Checkway". They focus on a full grammatical validation of messages by Checkway before forwarding them to the server. To do this, they consider that Web Service messages are XML documents and these are usually defined by an XML Schema, written in the XML Schema definition language-a grammar language for XML. Checkway generates an XML Schema from a Web Service description and validates all Web Service messages against this schema. The approach presents a centralized model oriented to detect concrete type of attack inside Web Services. An adaptive framework for the prevention and detection of intrusions was presented in [14]. Based on a hybrid focus that combines agents, data mining and fuzzy logic, it is supposed to filter attacks that are either already known or new. Agents that act as sensors are used to detect violations to the normal profile using the data mining technique such as clustering, association rules and sequential association rules. The anomalies are then further analyzed using fuzzy logic to determine genuine attacks so as to reduce false alarms. If an attack is being detected, a specific component will act to prevent the attack from happening. An approach to countering DDoS and XDoS attacks against web services is presented by [16]. The system carries out request message authentication and validation before the requests are processed by the Web Services providers. The scheme has two modes: the normal mode and the under-attack mode. A component called "operations provider" decides which mode the system works in. In the under-attack mode, the service requests need to be authenticated and validated before being processed. Since the system is constructed from web services, it can be formed and reconfigured easily. Finally, a recent solution proposed by [17] presents a Service Oriented Traceback Architecture (SOTA) to cooperate with a filter defense system, called XDetector. XDetector, is a Back Propagation Neural Network, trained to detect and filter XDoS attack message. SOTA is a traceback system that is constructed on the basis of Web Services and is able to traceback to the source of the malicious message. Once an attack has been discovered and the attacker's identity known, XDetector can filter out these attack messages.

Our approach outperforms the existing models with respect to:

- Learning and Adaptive ability: These features are the most important of our approach. Our approach includes one type of intelligent agents that was designed to learn and adapt to changes in attack patterns and new attacks.
- Tolerance to Failure: Our approach has a design that can facilitate error recovery through the instantiation of new agents.
- Scalability: Our approach is capable of growing (by means of the instantiation of new agents) according to the needs of its environment.

Our approach presents novel characteristics that have not heretofore been considered in previous approaches. The next section presents the architecture in greater detail. The following sections detail the internal model of the CBR-BDI agent, as well as the classification process for SOAP message for identifying malicious messages.

III. Classifiers Agents Internal Structure

Agents are characterized by their autonomy; which gives them the ability to work independently and in real-time

environments [18]. Because of this and their other capacities, agents are being integrated into security approaches such as Intrusion Detection Systems (IDS) [19]. However, the use of agents in these systems focuses on the retrieval of information in distributed environments, which only takes advantage of their mobility capacity.

The classification agent presented in this study interacts with other agents within the architecture. These agents carry out tasks related to capturing messages, syntactic analysis, administration, and user interaction. As opposed to the tasks for these agents, the classification agent executes a classification of SOAP messages in two phases that we will subsequently define in greater detail.

In our research, the agents are based on a BDI model in which beliefs are used as cognitive aptitudes, desires as motivational aptitudes, and intentions as deliberative aptitudes in the agents [8]. However, in order to focus on the problem of the SOAP message attack, it was necessary to provide the agents with a greater capacity for learning and adaption, as well as a greater level of autonomy than a pure BDI model currently possesses. This is possible by providing the classifier agents with a CBR mechanism [9], which allows them to "reason" on their own and adapt to changes in the patterns of attacks.

Case-based Reasoning (CBR) is a type of reasoning based on the use of past experiences [9]. The purpose of case-based reasoning systems is to solve new problems by adapting solutions that have been used to solve similar problems in the past. The fundamental concept when working with case-based reasoning is the concept of case. A case can be defined as a past experience, and is composed of three elements:

- A problem description which describes the initial problem.
- A solution which provides the sequence of actions carried out in order to solve the problem.
- The final state which describes the state achieved after the solution was applied.

A case-based reasoning system manages cases (past experiences) to solve new problems. The way in which cases are managed is known as the case-based reasoning cycle. These systems execute the CBR cycle which consists of four sequential steps: retrieve, reuse, revise and retain [9]. The method proposed in [20] facilitates the incorporation of case-based reasoning systems as a deliberative mechanism within BDI agents, allowing them to learn and adapt themselves, lending them a greater level of autonomy than what is normally found in a typical BDI architecture [21]. Accordingly, our Classifiers agents can reason autonomously and therefore adapt themselves to changes in the attack patterns. The case-based reasoning system is completely integrated within the Classifiers agents. These agents incorporate a "formalism" that is easy to implement, in which the reasoning process is based on the concept of intention. Intentions can be seen as cases, which have to be retrieved, reused, revised and retained. A direct relationship between case-based reasoning systems and BDI agents can also be established if the problems are defined in the form of states and actions.

Case: <Problem, Solution, Result>	**BDI agent**
Problem: initial_state	Belief: state
Solution: sequence of <action,[intermediate_state]>	Intention: sequence of <action>
Result: final_state	Desire: set of <final_state>

Our Classifiers agents implement cases as beliefs, intentions and desires which lead to the resolution of the problem. As described in [22], [23], each state of a CBR-BDI agent is considered as a belief, including the objective to be reached. The intentions are plans of actions that the agent has to carry out in order to achieve its objectives, which makes each intention an ordered set of actions. Each change from state to state is made after carrying out an action (the agent remembers the action carried out in the past, when it was in a specified state, and the subsequent result). A desire will be any of the final states reached in the past (if the agent has to deal with a situation that is similar to one from the past, it will try to achieve a result similar to the one previously obtained). The Classifiers agents used in our solution, use these concepts to define a case structure for DoS attacks in SOAP messages.

As previously mentioned, the classifier CBR-BDI agent is the core of the multi-agent architecture and is geared towards classifying SOAP messages for detecting attacks on web services. Figure 1 shows the classifier CBR-BDI agents in each phase of the mechanism of classification. These CBR-BDI classifier agents will be explained in detail in next section.

IV. Mechanism for the Classification of SOAP Message Attack

The CBR-BDI classifier agent presented in section 3 incorporates a case-based reasoning mechanism that allows it to classify SOAP messages. The mechanism incorporated into the agent approaches the idea of classification from the perspective of anomaly-based detection. In the specific case of SOAP messages, it manages a case memory for each service offered by the Web Service environment, which permits it to handle each incoming message based on the particular characteristics of each web service available. Each new SOAP message sent to the architecture is classified as a new case study object. Focusing on the problem that is of interest to us, we will represent a typical SOAP message which consists of a type of wrapping that contains an optional heading and a mandatory body of text with a useful message load, as depicted in figure 2.

Based on the structure of the SOAP messages and the transport protocol used, we can obtain a series of descriptive fields to consider. Based on this information, we can present a two-part strategy for executing the classification process:

A. First Phase of the mechanism of Classification

The main goal of this initial phase is to carry out an effective classification, but without requiring an excessive amount of resources and time. As a CBR strategy is used, it is necessary to define the case structure used by the classifiers CBR-BDI agents. The fields of the case are obtained from the headers of the packages of the HTTP\TCP-IP transport protocol. Table I shows the fields taken into consideration to describe the problem.

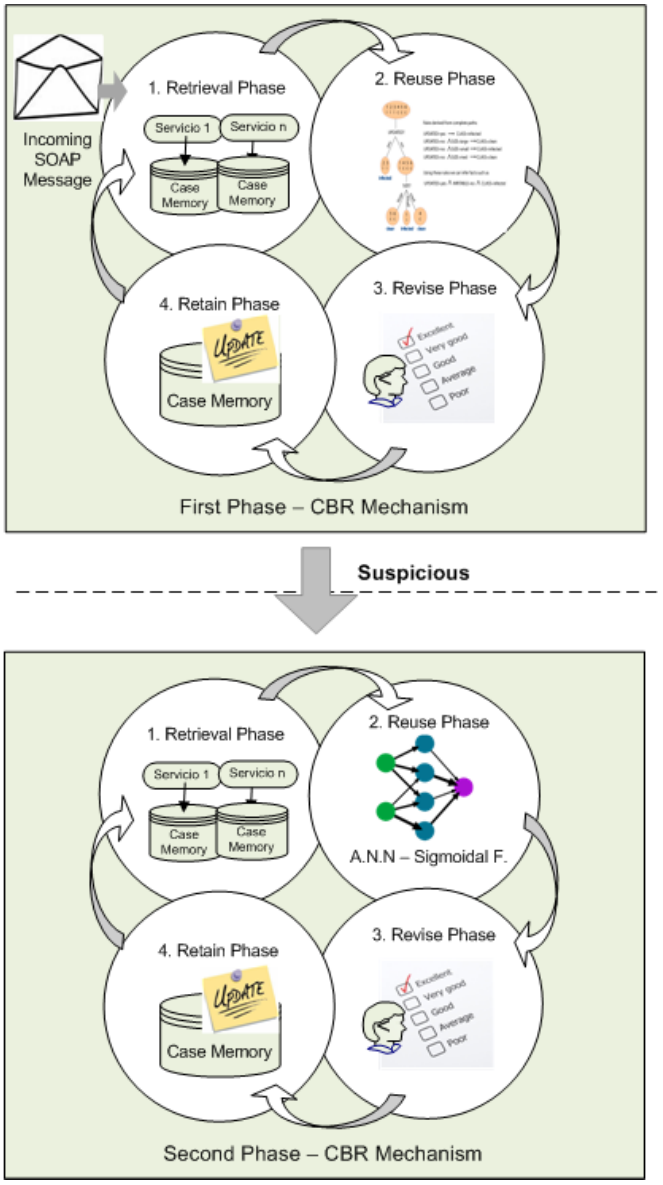

Fig. 1. Design of the mechanism of classification - Classifier CBR-BDI agents

Fig. 2. Content and Structure of a SOAP Message

TABLE I
CASE DESCRIPTION - CBR - FIRST PHASE

Fields	Type	Variable
IDService	Int	i
SubnetMask	String	m
SizeMessage	Int	s
NTimeRouting	Int	n
LengthSOAPAction	Int	l
TFMessageSent	Int	w

As can be seen in Table I, the description of a case is given by the tuple $c = (i, m, s, n, l, w, R/C_{.im}, x^p, x^r)$, where i represents the service identifier, m the subnet mask, s the message length, n the number of seconds for the travel of the message, l the length of the header SoapAction, w the elapsed time from the arrival of the last n messages, $R/C_{.im}$ is the solution provided by the decision tree associated to the service and to the subnet mask, x^p represents the class predicted by the CBR strategy $x^p \in X = \{a, g, u\}$, where a,g,u represent the values attack, good and undefined, x^r is the real class $x^r \in X = \{a, g, u\}$.

The CBR strategy is integrated into a BDI agent, obtaining a CBR-BDI agent. The integration of the CBR system and the BDI agent is defined as follows: beliefs - problem description and rules; intentions - set of accions and rules that represent the state transitions required to achieve the final state; desires - $X = \{a, g, u\}$. The initial state is defined by means of the set of beliefs that store the values for the subnet mask and the service web identifier, $(i, m, \phi, \phi, \phi, \phi)$. The intermediate states describe the decision process executed, taking into account the application of rules over the set of rules.

The cases memory contains a set of cases $C = \{c\}$ and is fragmented for each of the web services available in the server. This structure facilitates the depuration and analysis of the services in an independent manner. Separately to the cases memory, the agent incorporates a rules memory, constructed as a set of inductive rules defined as $R = \{r_1, \ldots, r_1\}$ with $r_i = (l_1 \wedge \ldots \wedge l_m) \rightarrow x_j$ where $l_s = (d_{ts}, o_s, \Re)/d_{ts} \in (i, m, s, n, l, w, x^p, x^r)$, $o_s \in O$, with $O = \{=, \neq, \succ, \prec, \leq, \geq\}$, $x_j \in X$. The rules memory is also fragmented for each of the services and for each of the subnet mask, in a way that $R/C_{.im}$ represents the rules associated to those cases belonging to the service i and the subnet mask m. For notation considerations, to identify a property of a case, we use the case, a point and the property. For example, $c_{j.m}$ represents the property m (subnet mask) of the case j.

When the agent receives a request to classify a new case c_{n+1}, a new execution of a CBR cycle is carried out. The following paragraphs describe the stages of a CBR cycle executed in the first phase classification.

- Retrieve: During this stage, those cases associated to the requested web service and the corresponding rules memory are retrieved. The storage and recovery of rules from the rules memory facilitates a notably reduction of the process time for the classification. The retrieve strategy is carried out as follows:
  - If there is not tree associated to the service and the subnet mask, then it is necessary to recover the cases for the service and the subnet mask:

$$C_{.im} = f_s(C) = \{c_{j.im} \in C / c_{j.i} = c_{n+1.i}, c_{j.m} = = c_{n+1.m}\} \quad (1)$$

    Where $C_{j.i}$ represents the case j and i the service identifier.
  - The rules memory associated with the set of cases $R/C_{.im}$ is retrieved
- Reuse: Knowledge extraction is especially important when complex algorithms that use hard computing techniques and that generate models in an automatic way are used. Human experts are much confident when they know exactly why or at least how a solution to a problem has been calculated. Classification And Regression Tree (CART) is a nonparametric statistical method for extraction of knowledge in classifications. The extracted information is represented in a binary decision tree, which allows individuals to be classified from the root node. Keeping the kind of dependent variable in mind, CART can be separated into two types: classification tree, if the dependent variable is categorical; and regression tree in the case of a continuous dependent variable.
  The reuse stage is only executed if not decision tree $R/C_{.im}$ associated to the cases $c_{.im}$ is available, and in order to do so, the rules are generated using the CART algorithm. $R/C_{.im} = CART(c_{.im})$ where $R/C_{.im}$ is the rules memory associated to the service identifier and to the subnet mask. The CART algorithm has been modified in order to have an automatic discretization of the values to a set of categories. The modification includes a first step to normalize the variable into the interval $[0, 1]$ and then, the values are discretized into one of the following categories depending on the closest value {very low=0.1, low=0.3, medium=0.5, high=, 0.7 y very high=0.9}. This way, the generation of rules using the CART algorithm is more efficient than working with a greater level of categories. The discretization is only carried out for the variables s, n, l, w.
- Revise: Once the set of rules has been retrieved, the classification for the case $c_{n+1.im}$ is obtained using the set of rules that previously classified the elements of the same type $c_{n+1.x^p = R/C_{.im}(c_{n+1})}$. If $r_i \in R/C_{.im}$ then, it is the rule that classifies c_{n+1}. The new case is classified as follows:
  - If $m_i \succ \mu_1 \| \#\{c_j \in C_{r_j} / c_{j.x^p} = u\} \succ \mu_2$ then, it is necessary to execute the CBR of the second phase. Where $C_{r_i} \subseteq C$ is the set of cases classified for r_i and m_i represents the percentage of misclassified cases of C_{r_j} using the rule r_j. # represents the number of elements of the set. The general idea is to verify if the error rate of the rule exceeds a certain threshold, and then, verify that the number of cases belonging to the set of elements classified using the rule not exceeds a certain threshold defined as a function of the total number of elements in C_{r_j}.
  - Else if $\#\{c_j \in C_{r_j} / c_{j.x^p} = g\} / \#C_{r_j} \succ \alpha_g$ then the case is classified as good and the revision finishes.
  - Else if $\#\{c_j \in C_{r_j} / c_{j.x^p} = u\} / \#C_{r_j} \succ \alpha_s$ the case is classified as suspicious and the second phase classification mechanism is executed.
  - Else if $\#\{c_j \in C_{r_j} / c_{j.x^p} = a\} / \#C_{r_j} \succ \alpha_a$ the case is classified as attack and the revision finishes.
- Retain: If the set of rules was generated because it didn't previously exist, then $R/C_{.im}$ is stored in the rules memory if the classification obtained was good. If the classification was erroneous and the misclassification was detected by an expert or if the second phase was invoked, then it is necessary to regenerate the decision tree: $R/C_{.im} = CART(c_{.im} \cup c_{n+1})$.

B. Second Phase of the mechanism of Classification

The fields are extracted from the SOAP message and provide the case description for second phase of the mechanism of classification. Table II presents the fields used in describing the problem for the CBR in this layer.

TABLE II
CASE DESCRIPTION - CBR- SECOND PHASE

Fields	Type	Variable
IDService	Int	i
SubnetMask	String	m
SizeMessage	Int	s
NTimeRouting	Int	n
LengthSOAPAction	Int	l
MustUnderstandTrue	Boolean	u
NumberHeaderBlock	int	h
NElementsBody	int	b
NestingDepthElements	int	d
NXMLTagRepeated	int	t
NLeafNodesBody	int	f
NAttributesDeclared	int	a
CPUTimeParsing	int	c
SizeKbMemoryParser	int	k

Applying the nomenclature shown in the table above, each case description is given by the following tuple:

$$c = (i, m, s, n, l, u, h, b, d, t, f, a, c, k, P/C_{.im}, x^p, x^r) \quad (2)$$

For each incoming message received by the agent and requiring classification, we will consider both the class that the agent predicts and the class to which the message actually belongs. x^p represents the class predicted by the classifier agent belonging to the group. $x^p \in X = \{a, g, u\}$; g and u represent attack, good and undefined, respectively; and x^r is the class to which the attack actually belongs, $x^r \in X = \{a, g, u\}$, $P/C_{.im}$ is the solution provided by the neural network Multilayer perceptron (MLP)associated to service i and subnet mask m.

The reasoning memory used by the agent is defined by the following expression: $P = \{p_1, \ldots, p_n\}$ and is implemented by means of a MLP neural network. Each P_i is a reasoning memory related to a group of cases dependent of the service and subnet mask of the client. The Multilayer Perceptron (MLP) is the most widely applied and researched artificial neural network (ANN) model. MLP networks implement mappings from input space to output space and are normally applied to supervised learning tasks [24]. The Sigmoidal function was selected as the MLP activation function, with a range of values in the interval $[0, 1]$. It is used to detect if the SOAP message is classified as an attack or not. The value 0 represents a legal message (non attack) and 1 a malicious message (attack). The sigmoidal activation function is given by:

$$f(x) = \frac{1}{1 + e^{-ax}} \quad (3)$$

The CBR mechanism executes the following phases:

- Retrieve: the cases that are most similar to the current problem, considering both the type of Web service to which the message belongs and the subnet mask that contains the message.

$$C_{.im} = f_s(C) = \{c_j \in C / c_{j.i} = c_{n+1.i}, \; c_{j.m} = c_{n+1.m}\} \quad (4)$$

Once the similar cases have been recovered, the neural network MLP $P/C_{.im}$ associated to service i and subnet mask m is then recovered.

- Reuse: The classification of the message is begun in this phase, based on the subnet mask and the recovered cases. It is only necessary to retrain the neural network when it does not have previous training. The entries for the neural network correspond to the case elements $s, n, l, u, h, b, d, t, f, a, c, k$. Because the neurons exiting from the hidden layer of the neural network contain sigmoidal neurons with values between $[0, 1]$, the incoming variables are redefined so that their range falls between $[0.2, 0.8]$. This transformation is necessary because the network does not deal with values that fall outside of this range. The outgoing values are similarly limited to the range of $[0.2, 0.8]$ with the value 0.2 corresponding to a non-attack and the value 0.8 corresponding to an attack. The training for the network is carried out by the error Backpropagation Algorithm [25]. The weights and biases for the neurons at the exit layer are updated by following equations:

$$w^p_{kj}(t+1) = w^p_{kj}(t) + \eta(d^p_k - y^p_k)(1 - y^p_k)y^p_k y^p_j + \\ +\mu(w^p_{kj})(t) - w^p_{kj}(t-1)) \quad (5)$$

$$\theta^p_k(t+1) = \theta^p_k(t) + \eta(d^p_k - y^p_k)(1 - y^p_k)y^p_k + \mu(\theta^p_k(t) - \\ -\theta^p_k(t-1)) \quad (6)$$

The neurons at the intermediate layer are updated by following a procedure similar to the previous case using the following equations:

$$w^p_{ji}(t+1) = w^p_{ji}(t) + \eta(1 - y^p_j)y^p_j(\textstyle\sum_{k=1}^{M}(d^p_k - y^p_k) \\ (1 - y^p_k)y^p_k w_{kj})x^p_i + \mu(w^p_{ji})(t-1)) \quad (7)$$

$$\theta^p_j(t+1) = \theta^p_j(t) + \eta(1 - y^p_j)y^p_j(\textstyle\sum_{k=1}^{M}(d^p_k - y^p_k) \\ (1 - y^p_k)y^p_k w_{kj}) + \mu(\theta^p_j(t) - \theta^p_j(t-1)) \quad (8)$$

where w^p_{kj} represents the weight that joins neuron j from the intermediate layer with neuron k from the exit layer, t the moment of time and p the pattern in question. d^p_k represents the desired value, y^p_k the value obtained for neuron k from the exit layer, y^p_j the value obtained for neuron j from the intermediate layer, η the learning rate and μ the momentum. θ^p_k represents the bia value k from the exit layer. The variables for the intermediate layer are defined analogously, keeping in mind that i represents the neuron from the entrance level, j is the neuron from the intermediate level, M is the number of neurons from the exit layer.

When a previously trained network is already available, the message classification process is carried out in the revise phase. If a previously trained network is not available, the training is carried out following the entire

procedure beginning with the cases related to the service and subnet mask, as shown in equation 9.

$$P_r = MLP^t(c_{.im}) \tag{9}$$

- Revise: This phase reviews the classification performed in the previous phase. The value obtained by exiting the network $y = P_r^e(c_{n+1})$ may yield the following situations:
  - If $y \succ \mu_1$ then it is considered an attack.
  - Otherwise, if $y \prec \mu_2$, then the message is considered a non-attack or legal.
  - Otherwise, the message is marked as suspicious and is filtered for subsequent revision by a human expert. To facilitate the revision, an analysis of the neural network sensibility is shown so that the relevance of the entrances can be determined with respect to the predicted value
- If the result of the classification is suspicious or if the administrator identifies the classification as erroneous, then the network repeats the training by incorporating a new case and following the BackPropagation training algorithm.

$$P_r = MLP^t(c_{.im} \cup c_{n+1}) \tag{10}$$

The next section presents the conclusions and results obtained of a developed prototype of our mechanism of classification.

V. Conclusion

This research has presented the nucleus of a novel solution that focuses on the protection of web services. The focus incorporates case-based reasoning methods, decision trees, fuzzy logic rules, neural networks, and intelligent agent technology that allows us to approach the problem of web security from a perspective based on learning, adaptability and flexibility.

The solution was designed to be carried out in two phases. In the first phase, a CBR mechanism incorporates decision trees; fuzzy logic rules generate a preliminary robust solution regarding the condition of the message, without sacrificing application performance. If the obtained solution is classified as suspicious, we then proceed to the second phase of the process. This phase does involve a more complex process, with a greater need for resources, and where a second CBR mechanism embeds within a neural network to generate a final result.

A prototype of our proposed solution was based on a classification mechanism and developed in order to evaluate its effectiveness. The tests of the simulation were carried out within a small web application developed with Java Server Page and hosted in a Apache Tomcat 6.0.18 Server by using as web service engine, Apache Axis2 1.4.1. The tests were organized within 6 blocks with a specific number of requests (50, 100, 150, 200, 250 and 300) that allowed evaluating the effectiveness of the classifier agent in accordance with the gained experience. Within the blocks were included legal and illegal requests. Figure 3 shows the results obtained.

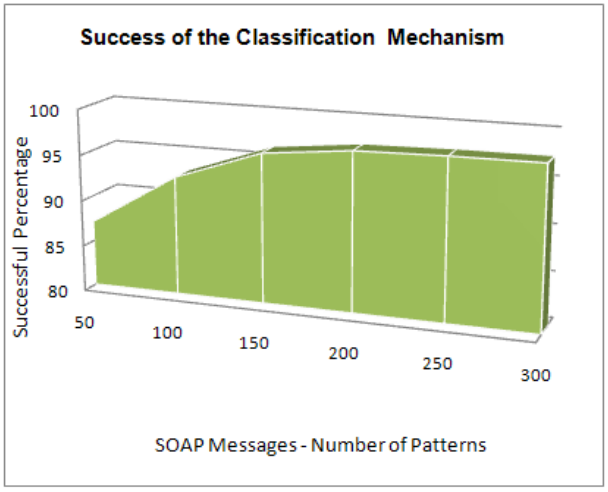

Fig. 3. Success of the Classification Mechanism

Figure 3 shows the percentage of prediction with regards to the number of patterns (SOAP messages) for the classification mechanism. It is clear that as the number of patterns increases, the success rate of prediction also increases in terms of percentage. This is influenced by the fact that we are working with CBR systems, which depend on a larger amount of data stored in the memory of cases.

The proposed solution will continue in the investigation and development for its application in various environments where its performance can be evaluated and real results obtained.

Acknowledgment

This development has been partially supported by the Spanish Ministry of Science project TIN2006-14630-C03-03 and The Professional Excellence Program 2006-2010 IFARHU-SENACYT-Panama

References

[1] M. A. Rahaman, A. Schaad, and M. Rits, "Towards secure soap message exchange in a soa," in *SWS '06: Proceedings of the 3rd ACM workshop on Secure web services*. New York, NY, USA: ACM, 2006, pp. 77–84.

[2] OASIS, "Web services security: Soap message security 1.1 (ws-security 2004)."

[3] G. Della-Libera, M. Gudgin, P. Hallam-Baker, M. Hondo, H. Granqvist, and C. Kaler, "Web services security policy language version 1.0 (ws-securitypolicy)," 2005.

[4] S. Anderson, J. Bohren, T. Boubez, M. Chanliau, G. Della, and B. Dixon, "Web services trust language (ws-trust)," 2004.

[5] S. Anderson, J. Bohren, T. Boubez, M. Chanliau, G. Della-Libera, and B. Dixon, "Web services secure conversation language (ws-secureconversation) version 1.1." 2004.

[6] N. Gruschka and N. Luttenberger, "Protecting web services from dos attacks by soap message validation," in *SEC*, 2006, pp. 171–182.

[7] R. Laza, R. Pavn, and J. M. Corchado, "A reasoning model for CBR_BDI agents using an adaptable fuzzy inference system," in *10th Conference of the Spanish Association for Artificial Intelligence*, ser. Lecture Notes in Computer Science, R. Conejo, M. Urretavizcaya, and J. L. P. de la Cruz, Eds., vol. 3040. Springer, 2003, pp. 96–106.

[8] A. S. Rao and M. P. Georgeff, "Modeling rational agents within a BDI-architecture," in *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, J. Allen, R. Fikes, and E. Sandewall, Eds. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1991, pp. 473–484. [Online]. Available:

[9] A. Aamodt and E. Plaza, "Case-based reasoning: foundational issues, methodological variations, and system approaches," *AI Commun.*, vol. 7, no. 1, pp. 39–59, March 1994.

[10] H. Bittencourt and R. Clarke, "Use of classification and regression trees (cart) to classify remotely-sensed digital images," in *Geoscience and Remote Sensing Symposium, 2003. IGARSS '03. Proceedings. 2003 IEEE International*, vol. 6, July 2003, pp. 3751–3753 vol.6.

[11] J. Shun and H. A. Malki, "Network intrusion detection system using neural networks," *International Conference on Natural Computation*, vol. 5, pp. 242–246, 2008.

[12] J. Snell, D. Tidwell, and P. Kulchenko, *Programming Web Services with SOAP*. O'Reilly, 2001.

[13] Y.-S. Loh, W.-C. Yau, C.-T. Wong, and W.-C. Ho, "Design and implementation of an xml firewall," *International Conference on Computational Intelligence and Security*, vol. 2, pp. 1147–1150, Nov. 2006.

[14] C. G. Yee, W. H. Shin, and G. S. V. R. K. Rao, "An adaptive intrusion detection and prevention (id/ip) framework for web services," in *International Conference on Convergence Information Technology (ICCIT '07)*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 528–534.

[15] M. Jensen, N. Gruschka, R. Herkenhoner, and N. Luttenberger, "Soa and web services: New technologies, new standards - new attacks," *Fifth European Conference on Web Services*, pp. 35–44, Nov. 2007.

[16] X. Ye, "Countering ddos and xdos attacks against web services," in *IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, vol. 1, 2008, pp. 346–352.

[17] A. Chonka, W. Zhou, and Y. Xiang, "Defending grid web services from xdos attacks by sota," in *IEEE International Conference on Pervasive Computing and Communications*, 2009, pp. 1–6.

[18] C. Carrascosa, J. Bajo, V. Julian, J. M. Corchado, and V. Botti, "Hybrid multi-agent architecture as a real-time problem-solving model," *Expert Syst. Appl.*, vol. 34, no. 1, pp. 2–17, 2008.

[19] A. Abraham, R. Jain, J. Thomas, and S. Y. Han, "D-scids: distributed soft computing intrusion detection system," *Journal of Network and Computer Applications*, vol. 30, no. 1, pp. 81–98, 2007.

[20] J. M. Corchado and R. Laza, "Constructing deliberative agents with case-based reasoning technology," *International Journal of Intelligent Systems*, vol. 18, pp. 1227–1241, 2003.

[21] M. E. Bratman, D. J. Israel, and M. E. Pollack, "Plans and resource-bounded practical reasoning," in *Computational Intelligence*, vol. 4, 1988, pp. 349–355.

[22] J. Bajo, J. F. D. Paz, D. I. Tapia, and J. M. Corchado, "Distributed prediction of carbon dioxide exchange using cbr-bdi agents," *International Journal of Computer Science*, pp. 16–25, 2007.

[23] J. M. Corchado, M. Glez-Bedia, Y. D. Paz, J. Bajo, and J. F. D. Paz, "Replanning mechanism for deliberative agents in dynamic changing environments," *Computational Intelligence*, vol. 24, pp. 77–107, 2008.

[24] M. Gallagher and T. Downs, "Visualization of learning in multilayer perceptron networks using principal component analysis," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 33, no. 1, pp. 28–34, Feb 2003.

[25] Y. LeCun, L. Bottou, G. Orr, and K. Muller, "Efficient backprop," in *Neural Networks: Tricks of the trade*, G. Orr and M. K., Eds. Springer, 1998.

Supporting Social Knowledge in Multiagent Systems through Event Tracing

Luis Búrdalo, Andrés Terrasa, Ana García-Fornes and Agustín Espinosa

***Abstract*—Social knowledge is one of the key aspects of MAS in order to face complex problems in dynamical environments. However, it is usually incorporated without specific support on behalf of the platform and that does not let agents take all of the advantage of this social knowledge. At present time, the authors of this paper are working in a general tracing system, which could be used by agents in the system to trace other agents' activity and that could be used as an alternative way for agents to perceive their environment. This paper presents first results of this work, consisting of the requirements which should be taken into account when designing such a tracing system.**

***Index Terms*—Agents, multiagent systems, social knowledge, tracing systems.**

I. Introduction

THESE days, the use and importance of multiagent systems (MAS) has increased because their flexible behavior is very useful to deal with complex problems in dynamic and distributed environments. This is not only due to agents individual features (like autonomy, reactivity or reasoning power), but also to their capability to communicate, cooperate and coordinate with other agents in the MAS in order to fulfil their objectives.

The necessary knowledge to support this social behavior is referred to by Mařik et al in [15] as *social knowledge*. This social knowledge plays an important role in increasing the efficiency in highly decentralized MAS. Social abstractions such as teams, norms, social commitments or trust are the key to face complex situations using MAS; however, these social abstractions are mostly incorporated to the MAS at user level; this is, from the multiagent application itself, without specific support from the multiagent platform, by means of messages among agents or blackboard systems. This weak integration of high level social abstractions, also mentioned by Bordini et al in [3], prevents agents in the MAS from exactly knowing what is happening in their environment, since they depend on other agents actively informing them about what they are doing. This dependance on other agents sets out two major problems. First, it can lead to excesive overhead in some of the agents. And second, it is also difficult to trust the information provided directly by other agents using messages in open MAS.

An alternative solution to provide social knowledge could be an event tracing system, integrated within the multiagent platform, which could be used by agents in the system to perceive their environment without having to actively notify each change to the rest of the agents which could be interested in what they do. Such a tracing system, integrated within the multiagent platform and providing a trustworthy event set which were capable to reflect not only communication among agents, but also agents' perceptions, etc, could be used as a way to provide social knowledge to the MAS. Also, such a tracing system would be more trustworthy than agent messages, since the information would not be proportioned by agents, but by the multiagent platform itself. Agents could trust the trace system as much as they can trust the multiagent platform.

Applications which extract information from the system by processing event streams at run time are already considered in the field of event driven architectures [14] and the idea of an standard tracing system available for processes in a system already existed in the field of operating systems (and at present it is contemplated by the POSIX standard[11]). These concepts can be applied to th field of MAS, where event tracing is still considered a facility to help MAS developers in the verification and validation processes.

This paper presents the requirements of such a general, platform-integrated tracing system applied to MAS. These requirements should be taken into account in order to develop a general abstract trace model for MAS, which could be finally incorporated to a real multiagent platform. The rest of the paper presents is structured as folows: Section II comments existing work by other authors in the field of tracing MAS. Section III presents a set of requirements which should be taken into account in order to design a general tracing system which could be used to improve agents sociability. Finally, section IV comments this work's main conclusions and future work which is still to be carried out in order to incorporate such a tracing system to a MAS.

II. Event tracing in multiagent systems

One of the most popular tracing facilities for MAS is the Sniffer Agent provided by JADE[1]. This tool keeps tracking of all of the messages sent or received by agents in the system and allows the user/administrator/developer to examine their content. These messages can be stored in a log file to be examined after the application has stopped running, so that the MAS can also be traced off-line. JADE also provides an Introspector Agent, which can be used to examine the life cycle of any agent in the system, its behaviors and the messages it has sent or received.

JADEX[18] provides a Conversation Center, which allows a user to send messages directly to any agent while it is

All authors are from:
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
cno./ de Vera S/N - 46022 Valencia (SPAIN)
E-mail: {lburdalo,aterrasa,agarcia,aespinos}@dsic.upv.es

executing and to receive answers to those messages from a user-friendly interface. It also provides a DF Browser to track services offered by any agent in the platform at run time and a BDI Tracer which can be used to visualize the internal processes of an agent while it is executing and show causal dependencies among agents' beliefs, goals and plans. Apart from these facilities, JADEX also incorporates a Remote Agent, which provides access to some of JADE's tracing facilities, like the Agent Introspector and the Sniffer Agent.

The JACK[20] multiagent platform does not provide a sniffer agent, but it supports monitoring communication among agents by means of Agent Interaction Diagrams. It also provides other introspecting tools with different functionalities: a Design Tracing Tool, to view internal details of JACK applications during execution, and a Plan Tracing Tool, to display and trace the execution of plans and the events that handle them. JACK also provides debugging tools that work at a lower level of abstraction in order to debug the multiagent system in a more exhaustive way: Audit Logging, Generic Debugging/Agent Debugging.

Other examples of tracing facilities provided by platforms is ZEUS' Society Viewer[10] which, apart from showing organisational inter-relationships among agents in the system, it can also show messages exchanged among agents. ZEUS also provides an Agent Viewer, which allows the user/administrator to monitor and change the internal state of the agent, its actions, used resources, etc. JASON[4], [5] provides its Mind Inspector Tool, to examine the internal state of agents across the distributed system when they are running in debugging mode.

Apart from those tools provided by multiagent platforms themselves, there are many tracing facilities provided by third party developers. This is the case of Java Sniffer[21], developed by Rockwell Automation, a stand alone java application based on JADE's Sniffer Agent which is able to connect to a running JADE system in order to track messages among agents, to reason about them and to show them to the user from different points of view. Another third party tool based on JADE's Sniffer Agent is ACLAnalyser[9], which intercepts messages interchanged by agents during the execution of the application and stores them in a relational database. After the execution, this message database can be inspected to detect social pathologies in the MAS. Later work by the same authors ([8]) combine results obtained with ACLAnalyser with data minning techniques to help in the MAS debugging process.

MAMSY, the management tool presented in [19] lets the system administrator monitorize and manage a MAS running over the Magentix multiagent platform[2]. MAMSY provides graphical tools to interact with the MAS and visualize its internal state at run time, including not only nodes and agents, but also organizational units. It also provides a message tracing tool, similar way to JADE's Sniffer Agent, which lets the system administrator visualize message interchange among agents.

In [16], the authors describe an advanced visualisation tools suite for MAS developed with ZEUS, although the authors also claim these tools could be used with other platforms (more precisely, with CommonKADS). The developed suite allows for inspecting message interchange among agents in a society, displaying graphically the different tasks in the society and its execution state, examining and modifying the internal state of any of the agents in the system and comparing statistics not only for individual agents, but also for agent societies. It also allows for the graphical display of the different tasks in the society and their execution states, examining and modifying the internal state of any of the agents in the system and comparing statistics not only for individual agents, but also for agent societies.

Tracking messages has also been used in [17], which comments an ampliation of the Prometheus methodology and the related design tool to help the designer to detect protocol violations by tracing conversations among agents in the system and to detect plan selection inconsistencies.

Lam et al present in [13] an iterative method based on tracing multiagent applications to help the user understanding the way those applications internally work. Lam et al also present a Tracer Tool which implements the described Tracing Method. The Tracer Tool can be applied to any agent system implementation, regardless of agent or system architecture, providing it is able to interface with Java's logging API (directly or via a CORBA interface). Results obtained with this method were presented in [12]. Bose et al present in [7] a combination of this Tracer Tool with a Temporal Trace Language (TTL) Checker presented in [6]. This TTL Checker enables the automated verification of complex dynamic properties against execution traces.

As it can be appreciated, tracing facilities in MAS are usually conceived as debugging tools to help in the validation and verification processes. It is also usual to use these tracing tools as a help for those users which have to understand how the MAS works. Thus, generated events are destinated to be understood by a human observer who would probably use them to debug or to validate the MAS and tracing facilities are mostly human-oriented in order to let MAS users work in a more efficient and also comfortable way. Some multiagent platforms provide their own tracing facilities, although there is also important work carried out by third party developers. However, even those tracing facilities which were not designed by platform developer teams are usually designed for a specific multiagent platform. There is not a standard, general tracing mechanism which let agents and other entities in the system trace each other as they execute like the one provided by POSIX for processes.

III. Tracing system requirements

From the viewpoint of the tracing process, a MAS can be considered to be formed by set of tracing entities, or components that are susceptible of generating and/or receiving tracing *events*. The tracing system needs to consider, at least, the following list of components inside the MAS as tracing entities: agents, organizational units (or any type of agent aggregation supported by the multiagent platform) and the multiagent platform itself (and its components).

Unlike existing work on tracing MAS, previously mentioned in Section II, a tracing system which could be used as a

knowledge provider must not be human-oriented, but entity-oriented, so that these tracing entities are able to receive events and process them or incorporate them to their reasoning process at run time in order to take advantage from that.

In order to generate trace events, the source code of tracing entities needs to be *instrumented* to include the code which actually produces such events. Attending to where this instrumentation code is placed, trace events can be classified as *platform events* or *application events*.

Platform events are instrumented within the source code of the platform (either in its "core" or in any of its supporting agents). These events represent the generic, application-independent information that the platform designer intends to provide to agents. On the other hand, application events are instrumented within the code of the application agents. These events represent customized run-time information defined by the application designer in order to support specific needs of the application agents.

The rest of the section presents a set of requirements which should be taken into account when developing such a tracing system. These requirements have been classified in three main groups: functional, efficiency and security requirements.

A. Functional requirements

Tracing roles. Any tracing entity in the MAS must be able to play two different roles in the tracing process: *event source* (*ES*) and *event receiver* (*ER*). From the viewpoint of tracing entities, these two tracing roles are dynamic and not exclusive, in the sense that each tracing entity can start and stop playing any of them (or both) at any time, according to its own needs. The relation between ES and ER entities is many to many: it must be possible for events generated by an ES entity to be received by many ER entities, as well as it must also be possible for an ER entity to receive events from multiple ES entities simultaneously.

Chronologically ordered event delivery requirement: Events generated in the system must be delivered to ER entities in chronological order or, at least, include information related to the time when they were produced to allow ER entities to process them in chronological order.

Dynamic definition of event types. Trace events can be classified in event types attending to the information which is generated and attached to them when they are generated. In order to let the event processing be more flexible and efficient, it must be possible for tracing entities to dynamically define new event types at run time. This must be applied to both platform and application event types.

Publication of event types. At any time, ER entities must be able to know which ES entities are producing events and of which types. So, as a consequence of event types being dynamic, the tracing system should keep and up-to-date list of such traceable event types (and ES entities) and to make this list available to all tracing entities in the MAS.

On-line and off-line tracing. In order to let entities work with both historical and run time information, both on-line and off-line tracing should be supported. In on-line tracing, events are delivered to ER entities as they are traced by the tracing system (with a potential delay due to the internal processing of events by the tracing system). In contrast, in off-line tracing, events generated by ES entities are not delivered to running entities, but stored in a log file. Both tracing modes must not be exclusive, meaning that it must be possible for the events generated by any ES entity to be delivered to some ER entities while also being stored in some log files. However, the tracing system does not need to support concurrent access to the events stored in a log file.

B. Efficiency requirements

In any computing system, tracing can be a very expensive process in terms of computational resources. In the case of MAS, the fact that they are by nature highly decentralized systems, both in number of running entities (agents) and hosts, can make their tracing even more expensive. In this context, the tracing process must be optimized in order to minimize the overhead it produces to the system, since a very sophisticated but excessively costly tracing system can become completely useless in practice. The following list introduces a minimal set of efficiency design guidelines that should be considered when designing a tracing system for MAS, in order to make this system realizable and useful. The first two requirements focus on the potential overload of the tracing system while the last one allows entities to set their own limits in the resources devoted to the tracing process.

Selective event delivery. Each ER entity should be able to express which event types it wants to receive, and the tracing system should only deliver events which belong to such types to the entity. Furthermore, each ER entity should be able to change dynamically which events it wants to receive, since entities may need different tracing information at different times during their execution.

Selective event tracing. The tracing system should not spend resources in tracing events which belong to event types that currently no entity wants to receive.

Resource limit control. Each ER entity should be able to limit the maximum amount of its resources to be allocated to receive events, both in on-line and off-line tracing modes. In on-line tracing, if there is some memory data object where events are delivered to until they are retrieved by the corresponding ER entity, then this entity should be able to define the maximum amount of memory devoted to such data object. In off-line tracing, the ER entity that sets the tracing up to the corresponding log file should be able to define the maximum size of the file.

C. Security requirements

Tracing in an open MAS has obvious security issues, since many of the events registered by the tracing system may

contain sensitive information that can be used by agents to take advantage from, or even to damage, the MAS. This scenario enforces the necessity of applying some security policy over the events that can be delivered to entities, specially if they are application entities. This policy can be materialized in many different ways, but in essence, it has to allow for the definition of security rules in the MAS that limit the availability of events to the *right* ER entities. The following list of requirements express a minimum set of restrictions by which it is possible to incorporate such security rules to the tracing system.

Authorization to ER entities. Each ES entity in the system must be able to decide which ER entities can receive the events that it generates. This can be accomplished by means of an authorization mechanism, provided by the tracing system, which can be used by ES entities to restrict the event types that are available to each ER entity. Such authorization rules must be dynamic, so that ES entities are able to modify the list of authorized ER entities corresponding to each event type at run time.

Supervisor entities. Situations where an entity must be able to access to other ES entity's events in order to fulfil its objectives, even though the ES entity does not agree with that, are very common in MAS. This can happen, for instance, in normative environments where an agent has to watch the other in order to verify that norms are not being violated and to apply the corresponding sanctions in case they are. The tracing system must also provide mechanisms to let an ER receive events generated by an ES without its authorization under some circumstances.

Delegation of authorizations. If an ER entity is currently authorized to be delivered events corresponding to certain event types, then this entity can delegate this authorization to other ER entities in the system; then, each of them can do so with other entities (potentially forming an *authorization tree*). At any node in the tree, the corresponding entity can add or remove delegations dynamically. If a delegation is removed, all the potential subsequent delegations (subtree) are also removed.

Platform entities authorization. By definition, the tracing system must be granted the authorization for all event types defined in the MAS, both at the platform and application levels. This is required for the tracing system to be able to keep track of any event being generated in the MAS, independently of the privacy rules defined by each ES entity.

IV. Conclusions and future work

Social knowledge is one of the most important features that make MAS appropiate to deal with complex problems in dynamic and distributed environments. The key to this is the capacity of agents to communicate and coordinate with other agents in the MAS in order to get their objectives. This capacity, though based on high level social concepts such as social commitments, trust, norms or reputation, is usually incorporated to the MAS at user level, using messages or blackboard systems, without support from the multiagent platform. This can produce too much overhead, reducing the scalability of the MAS. Also, it has to be taken into account that sometimes it is difficult to trust information from other agents, specially in open MAS.

A general event tracing system, which agents in the MAS could use to trace other agents in their environment, could be used as a more appropiate and trustworthy social knowledge provider. This paper presents the first step towards defining such a tracing system, which is the identification of its requirements. This paper has identified requirements in different aspects: functionallity, efficiency and security.

Some of the presented requirements set important problems out. Some of these problems are more obvious. For example, the problem of delivering events in chronological order in a distributed MAS. However, others are less evident. For instance, the problem of determining which ES entity is the owner of each trace event, since the instrumented code that produces an event is not always within the source code of the entity which originated it. Just as an example, consider events could as property of those entities which source code has been instrumented to produce them. In this case, all platform events would belong to the multiagent platform, while agents in the MAS would only be owners of application events. It could be more understandable and easier to incorporate considering that events belong to the ES entity which originated them.

Future work will include the design of a general abstract model for MAS which contemplated all of the requirements exposed above and which, after that, could be implemented and incorporated to a multiagent platform.

Acknowledgment

This work is partially supported by projects PROMETEO/2008/051, CSD2007-022 and TIN2008-04446, which is co-funded by the Spanish government and FEDER funds.

References

[1] U. AGREEMENT. Jade administrator's guide. *sharon.cselt.it*.

[2] J. Alberola, L. Mulet, J. Such, A. García-Fornes, A. Espinosa, and V. Botti. Operating system aware multiagent platform design. *Fifth European Workshop On Multi-Agent Systems (EUMAS 2007)*, pages 658–667, 2007.

[3] R. Bordini, M. Dastani, and M. Winikoff. Current issues in multi-agent systems development (invited paper). *Post-proceedings of the Seventh Annual International ...*, Jan 2007.

[4] R. Bordini and J. Hübner. Jason: A java-based interpreter for an extended version of agentspeak. page 31, Mar 2007.

[5] R. Bordini, J. Hubner, and R. Vieira. Jason and the golden fleece of agent-oriented programming. *MULTIAGENT SYSTEMS ARTIFICIAL SOCIETIES AND SIMULATED ...*, Jan 2005.

[6] T. Bosse, C. Jonker, L. van der Meij, and A. S. .... Specification and verification of dynamics in cognitive agent models. *... of the sixth international conference on intelligent agent ...*, Jan 2006.

[7] T. Bosse, D. Lam, and K. Barber. Tools for analyzing intelligent agent systems. *Web Intelligence and Agent Systems*, Jan 2008.

[8] J. Botia, J. Hernansaez, and A. Gomez-Skarmeta. On the application of clustering techniques to support debugging large-scale multi-agent systems. *Springer*, 2007.

[9] J. Botia, J. Hernansaez, and F. Skarmeta. Towards an approach for debugging mas through the analysis of acl messages. *MATES*, Jan 2004.

[10] J. Collis, D. Ndumu, H. Nwana, and L. Lee. The zeus agent building tool-kit. *BT Technology Journal*, Jan 1998.

[11] IEEE. *1003.1, 2004 EDITION IEEE Standard for Information Technology Portable Operating System Interface (POSIX)*. 2004.

[12] D. Lam and K. Barber. Debugging agent behavior in an implemented agent system. ... *Workshop on Programming Multi-Agent Systems at the Third* ..., Jan 2004.

[13] D. Lam and K. Barber. Comprehending agent software. *Proceedings of the fourth international joint conference on* ..., Jan 2005.

[14] D. Luckham. The power of events. *Addison-Wesley*, Jan 2002.

[15] V. Mafik and M. Pechoucek. Social knowledge in multi-agent systems. *Systems*, Jan 2004.

[16] D. Ndumu, H. Nwana, L. Lee, and J. Collis. Visualising and debugging distributed multi-agent systems. *Proceedings of the third annual conference on Autonomous* ..., Jan 1999.

[17] L. Padgham, M. Winikoff, and D. Poutakidis. Adding debugging support to the prometheus methodology. *Engineering Applications of Artificial Intelligence*, Jan 2005.

[18] A. Pokahr and L. Braubach. Jadex tool guide. page 66, Sep 2008.

[19] V. Sanchez-Anguix, A. Espinosa, L. Hernández, and A. García-Fornes. Mamsy: A management tool for multi-agent systems. *7th International Conference on Practical Applications of Agents and Multi-Agent Systems*, 2009.

[20] A. O. Software. Jack tm tracing manual. page 85, May 2008.

[21] P. Tichy and P. Slechta. Java sniffer 2.7 user manual. 2006.

A formal approach to test commercial strategies: Comparative study using Multiagent-based techniques

Luis F. Castillo, Manuel G. Bedia, Ana L Uribe, Gustavo Isaza

Abstract— **This paper presents a multiagent recommendation system (RecMAS) able to coordinate the interactions between a user agent (AgUser) and a set of commercial agents (AgComs). It provides a useful service for monitoring changes in the AgUser's beliefs and decisions based on two parameters: (i) the strength of its own beliefs and (ii) the strength of the AgComs' suggestions. The system was used to test several commercial activities in a shopping centre where the AgComs provided information to an AgUser operating in a wireless device (PDA, mobile phone, etc.) used by the client. The AgUser received messages adapted to the preferences of the client. Using a theoretical model and a set of simulation experiments, several commercial strategies were obtained. This paper concludes with a presentation of a prototype in a real shopping centre.**

Index Terms— **SocioConfiguration, Multiagent systems, Agent-based social simulation.**

I. INTRODUCTION

MODELS of artificial societies from different perspectives are useful in a large number of applications. Currently, there exist several different mathematical models that try to explain what types of relations are established in complex social systems [1]. Traditionally, theoretical models used to analyze complex social systems come from the field of social sciences, using qualitative techniques [2] but at present, new models from a quantitative perspective have started to be proposed [3]; in particular, models based in the theories of complexity [4] and emergent phenomena [5]. Nevertheless, although these models include very complex characteristics related with different domains, generally they do not take into consideration complex internal states. In this paper, Agent-based modelling is used to test formal models and techniques of the area of the multiagent systems are used for the design and implementation stage. Agent-based systems and multiagent systems [6,11] gather very interesting techniques in order to develop tools which can help us to describe (quantitatively) processes of change of beliefs and social adaptation.

The paper is divided in the following parts: section 2 provides a mathematical approach of the system describing the type of interactions between the agents in the proposed model. Briefly, there are *N* BDI-agents, one of which, the AgUser, is able to buy products in a virtual shopping centre stores adapted to a set of preferences. The *N–1* remaining agents, AgCom, propose changes in the beliefs of the AgUser involving purchase of new products.

In the section 3 is described how beliefs change due to social interaction and the relation between the importance of a belief and the social impact. It is shown how the AgUser's immediate beliefs depend on the position in the shopping area (i.e., based on the social interactions). Results are obtained by testing different strategies in a platform of agents based modelling (NetLogo).

In section 4 it is analyzed and showed the process of implementation. It is reproduced the set of behaviours found in the previous section by using Agent-oriented software engineering techniques. In particular, GAIA methodology for designing and analyzing stages, and 3-APL for the implementation on mobile devices of social phenomena that emerge in the system and different results obtained from the simulation of the system's behaviour.

The last two sections are faced to practical issues: section 5 is focused in aspects of optimizing and section 6 describes the results of a real trial in a shopping centre.

II. MULTIAGENT RECOMMENDATION MODEL

There are a huge amount of references on general simulation models. However, the problem and proposed model is focused on the simulation of shopper behavior in physical spaces [22,23,24,25]. In this more specific context, it will be proposed our own framework along this section.

Let's consider a 2-dimensional virtual shopping centre of area *A* where a community of *N* agents exists. N – 1 agents are AgComs which recommend the acquisition of products, and an AgUser which has "money" to buy them. In order to have a spatial description of the agents' system, it is assumed that the centre is divided in z sub-areas $A^* = A/z$ with *Nj* agents, *i.e.,* the number of agents in the sector *j*.

The number of the AgUser's beliefs, *m,* is shown below and the AgUser's belief concerning a problem is represented by the parameter αi.

$$A = \{\alpha_i\}_{i=1,\ldots..m} \qquad \alpha_i \in \{-r, \ldots., -1,0,1, \ldots., r\} \qquad (1)$$

The parameter αi can change from the interaction with other agents or new external information. It is assumed that negative values represent a pre*disposition to reject a proposed task*. Positive values represent a *predisposition for action*, and

values in the intermediate region of the interval, a *neutral position waiting for new information*.

A. Dynamic of the belief change in agents

This section starts with the equations for the dynamic of: (i) AgUser 's movements through the shopping centre and (ii) changes in preferences as a consequence of the interaction with several AgComs. It is suposed that: the AgUser starts the visit to the virtual shopping centre (area *A)* in a sub-area referred as k_0 (see figure 1a); it engages in purchases in sector *ks* with the distance between both positions (which coincides with the diagonal of the virtual environment) referred to as *D*(*A*); the AgUser can go to any of the places around him (8 places in the best case, see figure 1b); and, following the shortest route from *k* ,where it starts, to the final state *ks*. The route $\phi(k0, ks)$ is represented in figure (1a) which joins the "entrance sector" and the shopping sub-area. We refer as $\varphi(k, ks)$ the route that the AgUser would follow from another point *k* to *ks*. Supposing that AgUser is stated in the position *ki* and has got belief αi then the equation of the AgUser's movement -denoted as $\sigma(ki, kj)$-, from its actual position *ki* to one of the closer positions is as follows:

$$k_j \in V_{ki} \quad \wedge \quad k_i \rightarrow V_{ki} = \{ k_i^1, k_i^2, k_i^3, \ldots., k_i^8 \} \tag{2}$$

So,

- If the number of AgComs in *ki* with belief αi *(which coincides* with the AgUser's belief) is larger than the number of AgComs that share another belief αj with $j \neq i$, then the belief αi is reinforced and the AgUser chooses the path to sector *ks,* represented by,

$$\varphi(K, K_s) = K_s - K \tag{3}$$

- If the number of AgComs in *ki* with a different belief αj from αi is larger than that of those that share αi, the AgUser moves randomly to one of the nearest sectors kj Є V_{ki} in order to attain more beliefs. The formula that summarizes both of the AgUser's circulation strategies is shown below.

$$\sigma(k_i, k_j)_{|K_j \in V_k} = \begin{cases} k_j \quad \therefore K_j \in k_s - k, N_{K\alpha_j}^{K_i} \geq \sum_{j=1}^{m} N_{K\alpha_j}^{K_i} \\ Random\,(k_i, k_j), N_{K\alpha_j}^{K_i} < \sum_{j=1}^{m} N_{K\alpha_j}^{K_i} \end{cases} \tag{4}$$

Next, changes in the interaction strength of the agents in the AgCom environment are considered based on the AgUser's route. The previous section introduced two parameters: (i) *su* that measured the extent to which AgUser is inclined to maintain his belief αi, and (ii) *sc* that measured the influence the environment had over the AgCom's willingness to modify a belief αj. The parameter $\eta = (sc / su) \in (0, 1)$ measured the relation between both terms, (i) if $\eta = 0$, the AgUser do not change his beliefs as a result of the interaction with the AgCom, and (ii) if $\eta = 1$, the AgUser estimates his own beliefs equal to those he receives from the environment. Under such conditions it is expected that the influence of the environment on the AgUser who initially had a belief αi, to adopt a belief αj will be defined.

Then α_i^* is referred to as the value for AgUser belief αi , and α_i^n the value the agents AgCom attributes to that belief, where $n = 1,\ldots,N^k$. The influence needed to maintain the belief αi is determined by (5):

$$i_k(\propto_i) = s_u + \propto_i^* \sum_{n=1}^{N^k} s_c \,.\propto_i^n = s_u(1 + \propto_i^* \sum_{n=1}^{N^k} \eta \,.\propto_i^n) \tag{5}$$

The results obtained and the initial conclusions on how to improve AgCom strategies for attracting greater numbers of clients.

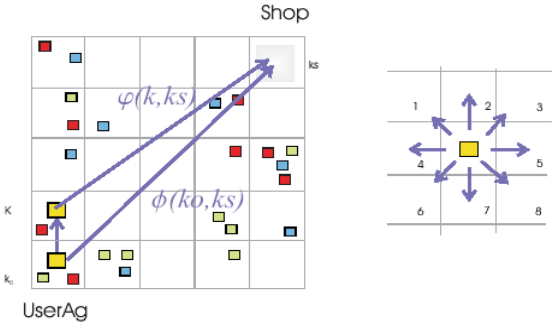

Fig. 1. (a) AgUser's movement in the shopping centre. (b) AgUser's possible movements

III. First Experiments with Agent Based Modelling Tools

In this section the first results obtained testing the previous mathematical model are showed. The analysis of these results enables to conclude ways to optimize the commercial strategies in three different experimental environments. A test-bed was used to find out the main features after checking several platforms in the field of agent based simulation. Those ones were: *NetLogo* [12], *MASON* [13,14], *Ascape* [15,16], *RePastS* [17,18], and *DIVAs* [19] given the fact that according to the literature, these are considered to be among the most effective platforms in the market [20]. After testing and reviewing the main features [21] (Environment structural complexity, Environment distribution, Agent and Environment coupling Specification offered by UI, Level of programming skill, Specified Environment knowledge in Agents, Quality of the visualization, Ease of change of properties of model and simulation view), our choice was NetLogo which enables the model simulation to be controlled by multiple users and distributed with respect to architecture and processing. NetLogo shows a very simple programming language to define, build and check models.

The following configurations were tested:

1) Commercial agents positioned near to the entrance. This strategy tries to persuade the user as soon as possible.
2) Commercial agents positioned close to area where the client pays for the product. This strategy tries to persuade the user when she/he is ready to pay.
3) Commercial agents situated across the diagonal D(A) and, therefore, with more chances to interact with the AgUser regardless of which route he/she does.

Each one of these environments corresponds to different strategies. For each of the configurations, it is monitored the evolution of parameter η of the AgUser (see previous section). The results can be seen in figure 2a, 2b and 2c.

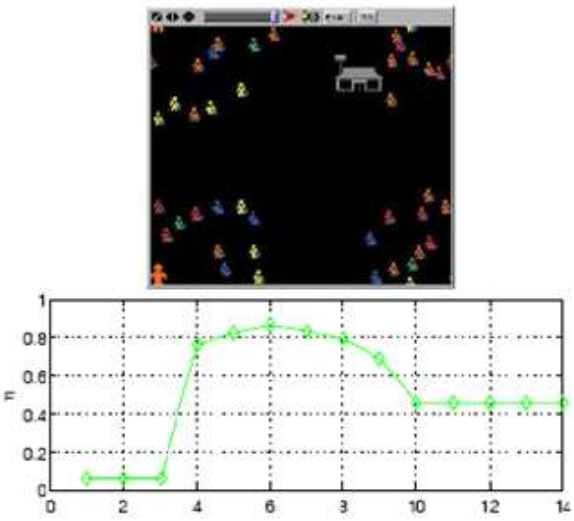

Fig. 2. (a). Commercial agents positioned near to the entrance

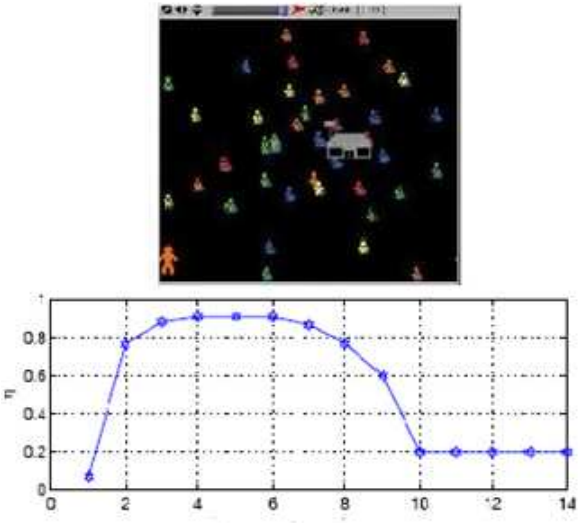

Fig. 2. (b). Commercial agents positioned close to the payment area

The results obtained seem to point to the following conclusions: the most useful strategy is related on distributions of agents able to interact with the UserAg *across the whole* way instead of waiting in strategic places.

In the next section some relevant issues related on the implementation of these models and strategies will be shown.

IV. Prototype and Implementation

The recommendation system in large shopping malls (RecMAS) is a computational tool based in MAS that attempts to convince clients to buy certain products. Each client, represented by an AgUser in a mobile device or PDA, interacts through a GPRS or wireless (Bluetooth or WiFi) connection with other agents in the shopping centre.

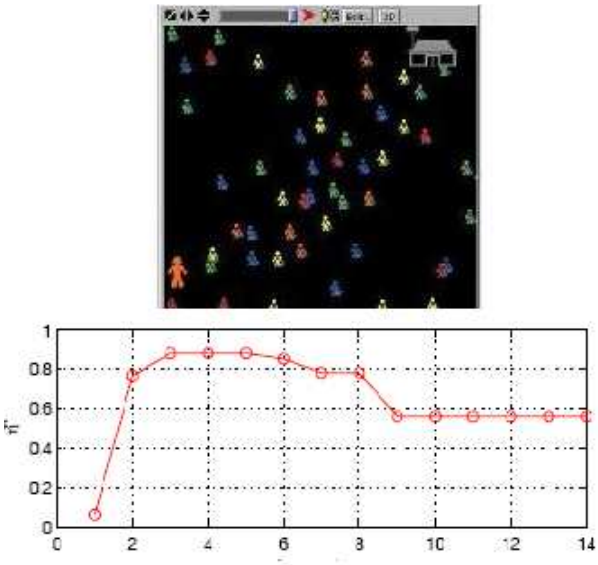

Fig. 2. (c). Commercial agents situated across the diagonal D(A)

Each store has *n* representatives (AgCom) distribted along the shopping centre with a scope limited to the sector *k* where they are located. From the point of view of implementation:

- The Gaia methodology [8] was used to model the multiagent system. It is comprised of role definition, protocols, services model, agents model and familiarity diagram. The Gaia methodology enables facile description of the agent system as an organization but has the inconvenience of not enabling a detailed level in the design stage. To correct this deficiency the modelling language Auml [9] was used. Auml is an extension of Uml (Unified Modeling Language) developed specially for agent (see figure 3 AgUser Class Diagram with AUML)
- Each agent has a deliberative architecture of the BDI sort implemented with a combination of its own platform and 3APL language [7]. 3APL (An Abstract Agent Programming Language) is a programming language for implementing cognitive agents. 3APL is based on a rich notion of agents, that is, agents have a mental state including beliefs and goals. 3APL language provides programming constructs for implementing agents' beliefs, goals, basic capabilities (such as belief updates, external actions, or communication actions) and a set of practical reasoning rules through which agents' goals can be updated or revised. The 3APL programs are executed on the 3APL platform. Each 3APL program is executed by means of an interpreter that deliberates on the cognitive attitudes of that agent.

As the AgUser's beliefs change after interaction with the AgComs in their coverage area, a procedure is defined for searching of nearby agents using a Bluetooth device. Figure 4 shows the deployment diagram application for mobile devices.

Some devices have experienced complications in sending/receiving processes when connecting via Bluetooth. In particular, Nokia 6620 and Nokia 6680 have worked successfully while the standard Bluetooth protocol JSR-82 has experienced inconsistent behavior. The reason appears to be

due to the multiplicity of j2me versions with different restrictions on hardware.

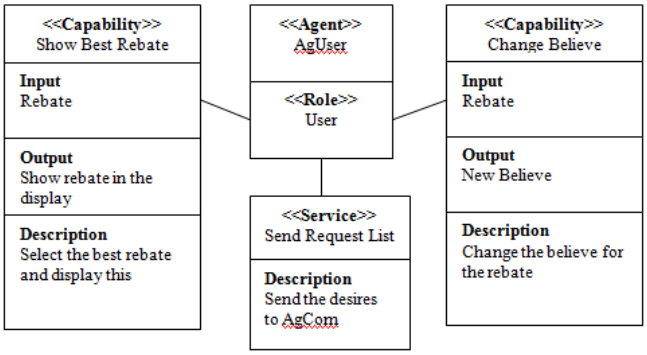

Fig. 3. AUML AgUser Class Diagram

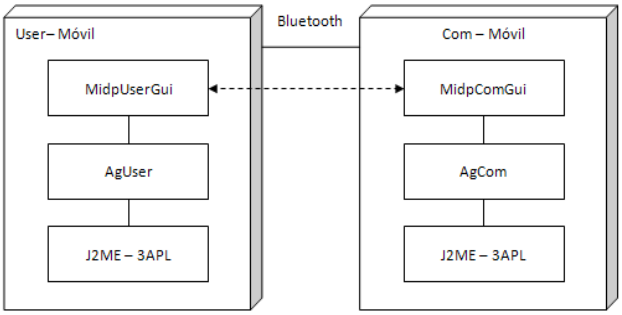

Fig. 4. Deployment Diagram for mobile device application

V. AGENTCOMS: TRADING KNOWLEDGE THROUGH LINKS

The following section seeks to optimize the interaction strategy of the AgCom, i.e., to calculate which percentage of AgComs move away from the diagonal or approach the beginning and end sectors, without decreasing the success level of the gradual strategy while attempting to attract the AgUser when it moves away from the diagonal path.

A. Complex optimization problem

From an analytical point of view complex *optimization problem* techniques and metrics should be used enabling characterization of a temporal AgCom distribution in two directions (*u, v*). It measures the AgComs' movement along the diagonal *D*(*A*) or anti-diagonal in the initial coordinates system (*x, y*) (See figure 5).

Movement inter-strategy: First, the progressive movement from a gradual strategy to environment type 2 and type 1 (Image 5A) are also calculated. The equations that determine such dynamics are determined by the following equations

$$\varphi_u^{(1)} = \frac{e^{\frac{-v^2}{2(\beta_0{}^2)}}}{2\pi\beta_0} \frac{e^{\frac{-u^2\ (1+\gamma t)^2}{2(\alpha_0)^2}}}{\alpha_0}(1+\gamma t)$$
$$\varphi_u^{(2)} = \frac{e^{\frac{-v^2}{2(\beta_0{}^2)}}}{2\pi\beta_0} \frac{e^{\frac{-(u-\frac{D(A)}{2})^2\ (1+\gamma t)^2}{2(\alpha_0)^2}}}{\alpha_0}(1+\gamma t) \quad (6)$$

Where $\varphi_u^{(2)}$ reflects the case in which the distribution towards the shopping sector with coordinates (D(A)/2,0) is studied; α y β being adjustable parameters:

$$\propto = \frac{\propto_0}{(1+\gamma t)}, \beta = \beta_0 \quad (7)$$

Spreading: Secondly, certain metrics are defined to characterize the level of AgCom distribution along the diagonal.

$$\varphi_v = \frac{e^{\frac{-u^2}{2(\propto_0{}^2)}}}{2\pi\alpha_0} \frac{e^{\frac{-v^2}{2(\beta_0-\gamma t)^2}}}{\beta_0+\gamma t} \quad (8)$$

Optimization of these two functions requires consideration of a multiobjective function where optimization of the distribution of the number of agents is in the directions *u* and *v* from the diagonal. The problem is very complex and inefficient; therefore an optimization model will be proposed based on time adjustment of a number of the AgCom positions: substituting calculus and computational cost through a communication based adaptation plan, (see formula 9, figure 5).

$$\phi_x^{(1)} = \frac{e^{\frac{-y^2}{2\beta_0^2}}}{2\pi\beta_0} \frac{e^{\frac{-x^2}{2\alpha_0^2}(1+\gamma^t)^2}}{\alpha_0}\ (1+\gamma^t) \quad (9)$$

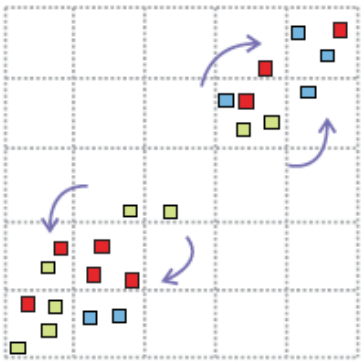

Fig. 5. Change of axis in the environment

B. Optimization using wireless communications

This section introduces a configuration model where rather than searching for a statistically successful static strategy, as in the previous example, the AgCom will follow simple coordination rules to adapt to AgUsers who are not following the diagonal. The proposed model shows how to find a dynamic adaptation to its environment in order to estimate if the AgUser is moving away from the diagonal route slightly or notably "as it goes". It is assumed that each AgUser can only communicate with other AgComs within their immediate environment. If the message sequence is constant, the distribution switches to the end sector (see figure 6(A)).

$$f_i(m+1) \propto f_{i-1}(m) + \gamma(u - \frac{D(A)}{2}) \quad (9)$$

If the message sequence is intermittent "the agent is close to the diagonal" and the distribution opens the range of values (see figure 6(B)).

$$f_i(m+1) \propto f_{i-1}(m) + \gamma(v - \frac{D(A)}{2}) \quad (10)$$

These simple rules force the AgComs to direct themselves towards (A), the shopping sector, or (B) to increase the range of their positions around the diagonal. The complexity of the communication relations based in these rules is not relevant (linear with regards to the number *N* of agents). The mobile devices are characterised by having less processing resources compared with computers. For this reason it was important to verify that the communication and search algorithms were not so complex that implementation would be impossible. A multithread routine was defined for the communication process that enabled interaction at a second level while information was presented to the user. One of the limitations of a Bluetooth Piconet is that it only supports up to 8 connected devices. In order to solve this restriction an JBAN library [10] was used enabling connection and control of an unlimited number N of devices in the network.

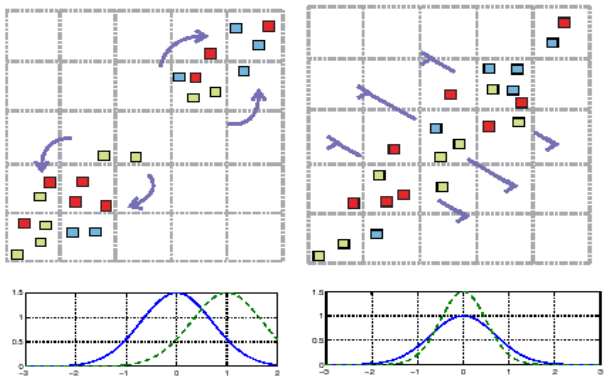

Fig. 6. Adaptation strategies: (A) Inter-strategy movement, (B) "Spreading" of a gradual distribution

VI. Final Results: Strategies for the Adaptation in Real Time and the Dynamic Attraction of Clients

One of the aspects taken into consideration for the evaluation of results was to gradually increase the area of scope. Initially 8 subregions in the shopping centre were defined. All of them shared the same vertex (entrance of the user) and as the subregion was extended, it contained the previous region. The size ratio between one subregion and the next is based on 40 meters, thus, the smallest is that size and the largest is 8 times bigger (the enlargement is mainly horizontal, because this shopping centre is not square). Different valuations were undertaken (30 users, three times each). In each case the belief they had initially when they entered the shopping centre and the one they bought the product were evaluated (see figure 7). The evaluated strategies where the: (i) "Lying-in-wait" strategy (yellow in the graph), (ii) Gradual strategy (crimson in the graph) and (iii) Adaptive strategy (blue in the graph). These results were as follows:

1. "Lying-in-wait" strategy: For smaller environments the level of success is higher that in larger environments: on average the AgUser covers more space and has more interactions with other AgComs.
2. "Gradual" strategy: The level of success is higher than in the previous strategy for all the environments and decreases with the size in a trend similar to the previous case. Nevertheless, what is interesting about the result is that in relative terms the slope of the decrease of success falls more quickly.
3. "Adaptive" strategy: The most interesting result according to the gathered information is that the relative degree in which the success of the strategy decreases is much smaller than in the previous examples. In other words, the "adaptive" strategy is the only one that has acceptable results when the dimensions of the space considered is of a surface of 300 m2 or higher.

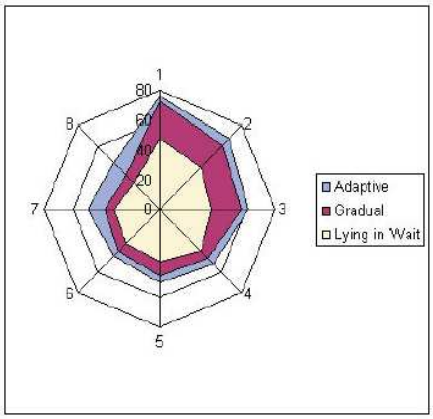

Fig. 7. Graphical representation of the level of the strategies' success: (A) "Lying-in-wait" strategy, (B) Gradual strategy (C) Adaptive strategy.

A set of 1000 simulations were developed and data obtained are shown in the figure 8. The messages in the exchange protocol between AgComs are represented by JnMensajes and nMensajes when the protocol was measured by optimizing of complex problem.

	nMensajes	JnMensajes
Count	1000	1000
Average	5,984	16,149
Variance	6,97021	140,926
Standard deviation	2,64012	11,8712
Minimum	1,0	1,0
Maximum	10,0	40,0
Range	9,0	39,0

Box-and-Whisker Plot

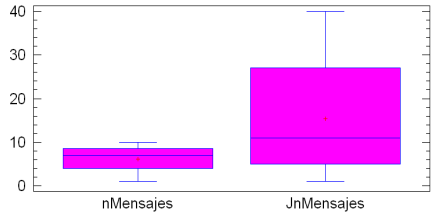

Fig. 8. (i) nMensajes in the optimizing problem and JnMensajes by TradeAgent protocol.

VII. Conclusions and Future Work

A model of relations between an AgUser and AgComs inside a multiagent society was developed. This model was designed to represent the internal strength of an AgUser's beliefs and its resistance to belief change processes affecting the products it was predisposed to buying. Next, different environments were proposed representing different advertising strategies that the AgCom used in large areas. The tests, carried out in a real environment, helped to state the advantages one environment enjoyed with the help of wireless technology over another environment. A wireless system was provides to the system user, represented as an AgUser. The mobile device supported Bluetooth, GPRS and j2me. The AgCom had PDA devices that supported Bluetooth, j2me and WiFi. Initially each AgUser stored its user profile, where its initial preferences were identified, and initiated the shopping centre route, visiting stores while being informed of new offers. Once they finished their route, they went to a sector store and bought the product indicated by their beliefs. The evolution of AgUser beliefs was registered according to the interaction with the AgComs. The concluding part of the paper suggests that a dynamic search strategy be sought seeking AgUser interaction taking advantage of the possibilities provided by the use of mobile devices. The communication model based on proximity interaction relations avoids network overloads by the distribution of AgComs over the surface (compared with GPRS technologies that enable the communication and identification of the whole group of existing agents but whose interaction flux is more complex). The result has been very satisfactory: the local scope and the simple rules of interaction have enabled discovery of a strategy that evolves parallel to the AgUser's movement. The AgCom changed from occupying a static location that responded to a general strategy, to auto-organization based on the messages they receive from their nearest neighbours, anticipating the movements of the AgUser in relation to the diagonal.

References

[1] Durfee E.H. (1998),Designing Organizations for Computational Agents, in Simulating Organizations: Computational Models of Institutions and Groups, Prietula, M. , Carley, K. and Gasser, L eds., Menlo Park, CA: AAAI Press and MIT Press, 1998.

[2] Gilbert, N. and Doran, J. (eds.) (1994). Simulating Societies: The computer simulation of Social processes. Londos. University College.

[3] Helbing, D. Quantitatuve Sociodynamics (1995). Stochastic Methods and Models of Social Interaction Processes. Dordrecht, Kluwer Academic.

[4] Weidlich, W. (1991). Physics and social science - The approach of synergetics. Phys. Rep. 204, 1-163.

[5] Weidlich, W. (1994). Synergetic modelling concepts for sociodynamics with application to collective political opinion formation. J. Math. Sociol. 18, 267-291.

[6] Epstein, J.M. and Axtell, R. (1996). Growing Artificial Societies: Social Science from the Bottom Up. MIT Press/Brookings Institution Press.

[7] Koch, F, Meyer John-Jules, Dignum Frank, Rahwan Iyad. (2005) Programming Deliberative Agents for Mobile Services: the 3APL-M Platform. AAMAS'05 Workshop on Programming Multi-Agent Systems (ProMAS05)

[8] Wooldridge M., Jennings, N. R., and Kinny, D., (2000) The Gaia Methodology for Agent-Oriented Analysis and Design, Journal of Autonomous Agents and Multi-Agent Systems, Vol. 15 2000.

[9] Berhard Bauer , J.P.M., James Odell, (2001) Agent Uml: A Formalism For Specifying Multiagent Interaction. Agent-Oriented Software Engineering, 2001 (springerverlag, berlin): p. 91-103.

[10] JBAN Library. https://jban.dev.java.net/

[11] Corchado J. M. , Gonzalez-Bedia M. , De Paz Y. ,Bajo J. y De Paz J.F. Replanning mechanism for deliberative agents in dynamic changing environments. Computational Intelligence ISSN: 0824-7935. Volumen 24, núm. 2, Pág. 77-107.2008

[12] U. Wilensky, NetLogo, http://ccl.northwestern.edu/netlogo/ Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

[13] MASON, http://cs.gmu.edu/_eclab/projects/mason/ George Mason University

[14] S Luke, C Cioffi-Revilla, L Panait, and K Sullivan, "MASON A New Multi-Agent Simulation Toolkit, Department of Computer Science and Center for Social Complexity", In Proceedings of SwarmFest, Michigan, USA, 2004

[15] Ascape, http://ascape.sourceforge.net/index.html

[16] Ascape documentation, http://ascape.sourceforge.net/index.html/#Documentation

[17] RePastS, http://repast.sourceforge.net/ (Argonne National Laboratory Decision and Information Sciences Division Center for Complex Adaptive Agent Systems Simulation).

[18] RePast3, http://repast.sourceforge.net/repast_3/index.html

[19] R.Z. Mili, R Steiner, E Oladimeji, "DIVAs: Illustrating an Abstract Architecture for Agent-Environment Simulation Systems", Multi agent and Grid Systems, Special Issue on Agent-oriented Software Development Methodologies 2 (4), 2006.

[20] S.F Railsback, S.L Lytinen, and S.K.Jackson, "Agentbased Simulation Platforms: Review and Development Recommendations", Simulation, vol. 82, 2006

[21] Arunachalam, S.; Zalila-Wenkstern, R.; Steiner, R.; Environment Mediated Multi Agent Simulation Tools – A Comparison.Self-Adaptive and Self-Organizing Systems Workshops, 2008. SASOW 2008. Second IEEE International Conference on 20-24 Oct. 2008 Page(s):43 – 48

[22] Schwaiger, A., Stahmer, B. SimMarket: Multiagent-based customer simulation and decision support for category management". Lecture Notes in Artificial Intelligence, 2831, pp. 74-84, 2003

[23] Turner, A., Penn, A. "Encoding natural movement as an agent-based system:an investigation into human pedestrian behaviour in the built environment". Environment and Planning B: Planning and Design, 29, pp. 473-490, 2002.

[24] Ali, W., Moulin, B. "2D-3D Multiagent GeoSimulation with knowledge-based agents of customers' shopping behavior in a shopping mall". Lecture Notes in Computer Science, 3693, pp. 445-458, 2005.

[25] Babin, B.J., Darden, W.R., Griffin, M. "Work and/or fun: measuring hedonic and utilitarian shopping value". Journal of consumer research 20(4), pp. 644-656, 1994.

[26] Guan, Y.; Ghose, A.K.;Executable specifications for agent oriented conceptual modelling. Intelligent Agent Technology, IEEE/WIC/ACM International Conference on 19-22 Sept. 2005 Page(s):475 – 478.

Norm Emergency through Argumentation

S. Heras, N. Criado, E. Argente and V. Julián

***Abstract*—Open societies are situated in dynamic environments and are formed by heterogeneous autonomous agents. In order to ensure social order, norms have been employed as coordination mechanisms. However, the dynamical features of open systems may cause that norms loose their validity and need to be adapted. Therefore, this paper proposes a new dialogue game protocol for modelling the interactions produced between agents that must reach an agreement on the use of norms. An application example has been presented for showing both the performance of the protocol and its usefulness as a mechanism for managing the solving process of a coordination problem through norms.**

***Index Terms*—Norm Emergency, Dialogue Games, Normative MAS**

I. Motivation

NOWADAYS, Multi-agent Systems (MAS) research on addressing the challenges related to the development of open distributed systems is receiving an increasing interest [8]. The main features of open systems are: (i) they are populated by heterogeneous agents which can enter or leave the system dynamically; and (ii) they are situated in dynamic environments. Therefore, their entities might be unknown and none assumption about their performance can be done [2]. As a consequence, mechanisms for coordinating their behaviours and ensuring social order are essential. In this sense, social factors are becoming more and more important to coordinate interactions in dynamic open worlds. Works on Normative Theory have been applied into the MAS area as a mechanism for facing up with undesirable and unexpected behaviours [7]. These regulatory mechanisms attempt to guarantee a globally efficient coordination in open systems taking into account the impossibility of controlling (the majority of) the agents and services directly.

Two different approximations have been considered as alternatives to the definition of norms: (i) off-line design, where the system designer defines the normative system statically [15]; and (ii) automatic emergence, which analyses how norms can emerge inside a group of agents [16]. The latter is more suitable for open systems, in which structural, functional and environmental changes might occur [17]. Therefore, dynamical situations may cause that the norms that regulate an organization lose their validity or should be adapted. In this second case, techniques for reaching an agreement among agents on the employment of norms are needed.

This research is aimed at providing a mechanism for managing norm emergence in open environments. THOMAS [4], [5], a development architecture for Virtual Organisations (VO) [3], has been selected as a suitable environment to test this proposal. The main idea that inspired this architecture was to give support a better integration between the standardised service-oriented computing technologies and the MAS paradigm. Both technologies can complement the strengths of each other: (i) service standards provide an infrastructure for the interaction among agents; (ii) MAS offer a more general and complex notion of *Service Oriented Architectures* (SOA); and (iii) intelligent and social capabilities of agents allow defining complex services. In THOMAS, the coordination among heterogeneous agents is achieved by means of norms. Thus, a normative language for formalising constraints on agent behaviours has been developed [1].

Our approach is to apply dialogue games as an argumentation technique to model the interaction among agents that must reach an agreement about the normative context. Dialogue games are interactions between players where each one moves by advancing locutions in accordance to some pre-defined set of rules [11]. In MAS, they have been used to specify communication protocols [10] and to evaluate reputation [12]. However, the application of argumentation techniques to the definition of norms is a novel area of research.

This paper is structured as follows. Section 2 briefly introduces background of this work; the THOMAS architecture and the dialogue game protocol on which the work is based. Section 3 describes the dialogue game protocol specification. Section 4 shows an example of the protocol applied to solve a coordination problem in THOMAS. Finally, section 5 summarises the main conclusions drawn from this research.

II. Background

This section briefly introduces the THOMAS architecture and the dialogue game adapted to cope with the objective of our research.

A. THOMAS Architecture

THOMAS architecture [4], [5] has been proposed for the generation of Virtual Organisations (VO) in open environments. The main idea that inspired THOMAS architecture was to achieve a better integration between the standardised service-oriented computing technologies and the MAS paradigm. Therefore, every operation that can be performed in the architecture (even the THOMAS management functionalities) is described and offered by means of Web Services standards. A description of the THOMAS architecture can be found at [1].

The *THOMAS architecture feeds on the FIPA*[2] architecture, but extends its main components —the Agent Management System (AMS) and the Directory Facilitator (DF) — into

S. Heras, N. Criado, E. Argente and V. Julián are with Universidad Politécnica de Valencia.
E-mail: sheras, ncriado, eargente, vinglada@dsic.upv.es

[1] http://www.fipa.org/docs/THOMASarchitecture.pdf
[2] http://www.fipa.org

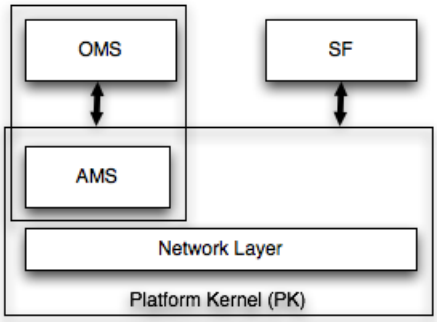

Fig. 1. Thomas architecture

an *Organization Management System* (OMS) and a *Service Facilitator* (SF), respectively. More specifically, the main components of THOMAS architecture are (see Figure 1):

- *Platform Kernel* (PK), that deals with basic agent management services and it can be provided by any FIPA-compliant platform. Its functionality is related with the agent life-cycle and the network communication layer.
- *Service Facilitator* (SF), which is a service manager that registers services provided by external entities and facilitates service discovering for potential clients. The SF can be considered as a yellow pages server.
- *Organization Management System* (OMS), which is responsible of the management of virtual organizations, taking control of their underlying structure, the roles played by the agents inside the organization and the norms that rule the system behaviour.

Open MAS require mechanisms for controlling agents behaviours and ensuring social order. The THOMAS architecture proposes the use of a normative system to cope with this requirement. Thus, the THOMAS architecture allows defining norms that prescribe agent rights and duties in terms of who can provide a service, when and under which circumstances. With this aim, a normative language for formalising constraints on agent behaviours has been developed [1]. This language allows the specification of norms that define deontic prescriptions to control access to services in THOMAS.

However, as a consequence of the dynamicity of the environments that THOMAS is intended for developing and the heterogeneity of their agents, changes in the normative context frequently occur. Thus, a mechanism for adapting the normative context to the current state of the VO is desirable. This adaptation process can be formalised as a general coordination problem. In [9], coordination is defined as 'managing dependencies between activities'. Therefore, a coordination problem arises when a group of agents share a goal which fulfilment requires the cooperation among the activities performed by agents [14]. In this research, the problem to cope with is to reach an agreement about the definition of a normative context. The interaction among the agents that participate in the agreement process is coordinated by means of a dialogue game protocol.

B. Argument from Expert Opinion

We have adapted a general dialogue game, the *Argumentation Scheme Dialogue (ASD)* [13] to formalise the interaction among agents that argue about a normative context. Our work is based on this general dialogue game, which extends traditional dialogue games with certain stereotyped patterns of common reasoning called *argumentation schemes*. Concretely, we have instantiated this game by making use of a specific argumentation scheme, the *Argument from Expert Opinion* [18] that captures the way in which people evaluates the opinions (recommendations about norms in our context) of experts (agents with some knowledge about a set of norms to recommend). The structure of the scheme is the following:

- *Major Premise:* Source E is an expert in field F containing proposition A.
- *Minor Premise:* E asserts that proposition A (in field F) is true (false).
- *Conclusion:* A may plausibly be taken to be true (false).

Moreover, this scheme has also a set of *critical questions*, which represent the possible attacks that can be made to rebut the conclusion drawn from the scheme and hence, are very useful for coordinating the dialogue:

1) Expertise: How credible is E as an expert source?
2) Field: Is E an expert in the field F that A is in?
3) Opinion: What did E assert that implies A?
4) Trustworthiness: Is E personally reliable as a source?
5) Consistency: Is A consistent with what other experts assert?
6) Backup Evidence: Is E's assertion based on evidence?

III. Dialogue Game Protocol Specification

In this section, we explain the main features of the dialogue game protocol proposed. First, the social structure that allows agents to evaluate arguments of other agents is shown. Based on it, the dialogue game protocol is specified. Finally, how agents can pose arguments and rebut attacks is also described.

A. Social Structure

Following the normative emergence approach, in this paper we propose the use of a service of normative assistance to solve coordination problems in VOs. This service is provided by a set of *Normative Assistant (NA)* agents that recommend the appropriate modifications to fit the operation of a VO to the current situation by adapting its normative context. The normative assistance services are published by the THOMAS SF and thus, they can be publicly requested by all agents of the organization.

Therefore, whenever an agent of the system wants to solve a coordination problem, it assumes the role of *initiator* and requests the SF a list of providers of the *normative assistance service*. Then, among these providers, it may select a subset of NAs with agents that it personally considers as 'friends' (previously known agents, if any). This friendship relation comes from past recommendation processes where the initiator was involved. The experience-based friendship relations of all agents of the system can be represented by a social network

abstraction. The network topology would be implicitly defined through the confidence relations that an agent has with its friends and is, thus, decentralised. In this network, nodes would represent agents and links would be friendship relations between them (labelled with confidence values).

Therefore, since the first recommendation dialogue where an agent was engaged in as *initiator*, this agent keeps a list of all the agents that participated in the dialogue and its final assessment about the confidence about the recommendations received from other agents. This *confidence degree* $c_{ij} \in [-1, 1]$ is updated at the end of each recommendation process by using a discrete value $u_j \in \{-1 : inappropriate,\ 0 : useless\ or\ 1 : useful\}$ that stands for the final usefulness of each recommendation received by the initiator. In addition, the initiator also informs the NAs that participated in the dialogue of the usefulness value of their recommendations. With this value, each NA can update its *expertise degree* e_j as norms recommender. These degrees are computed by using equations 1 and 2:

$$c_{ij} = \frac{\sum_{k=1}^{K} u_{j(k)}}{K} \quad (1)$$

$$e_j = \frac{\sum_{i=1}^{I} c_{ij}}{deg^{+}(a_j)} \quad (2)$$

In the equations, $u_{j(k)}$ is the usefulness degree of the recommendation k that an agent accepted from its friend a_j, K is the total number of recommendations accepted by an agent and $deg^{+}(a_j)$ is the *centrality indegree* of a_j (the number of agents that consider this agent as friend). Note that, although confidence and expertise degrees could be considered subjective and could be risky for an agent to believe them by default, these measures are defeasible and the interaction protocol explained in the next section allows agents to argue about them.

We also assume that each NA agent stores the set of norms that were effective to solve a coordination problem in the VOs where it has participated in a *norm database*. In addition, for every set of norms, it also stores the attributes that characterised the type of problem addressed. Therefore, the knowledge about the solving process of the different coordination problems is distributed across the network and each NA agent only has a partial view based on its own experience. Moreover, if a NA is asked for a set of norms to deal with a coordination problem that it has never been faced with, the agent can also propagate the query to its neighbours in the network by using its friendship relationships (i.e. such NA agents that have eventually provided this NA agent with recommendations in the past). When all NAs have made their proposals, the initiator has been presented with several recommendations about sets of norms. Then, it selects the best proposal by using the proposed dialogue game protocol.

Finally, note that each agent is assumed to have its own reasoning mechanism for evaluating preferences, matching them with its norm database and proposing recommendations. In addition, agents must also know a set of inference rules and the scheme from expert opinion to be able to create arguments. The definition of the individual reasoning mechanisms of agents (e.g. how they manage arguments and evaluate the usefulness degree of recommendations) is out of the scope of this paper. We have mainly focused here on formalising the dialogue-based interaction protocol. Following, the specification of this interaction protocol is detailed.

B. Protocol Specification

The dialogue game protocol proposed in this paper is an application of the ASD game instantiated with the Argument from Expert Opinion scheme presented in [6]. On one hand, the protocol assumes the existence of:

- A finite set of players denoted *Agents*, with elements $\{Ag_i, NA_1, ..., NA_n\}$, consisting of the agent Ag_i that plays the *Initiator* role and starts the dialogue and the set of normative assistants $\{NA_1, ..., NA_n\}$ in its social network that play the role of recommendations *proponents*.
- A finite set of recommended items, denoted *Items*, with elements $\{i_1, ..., i_m\}$. Each item consists of the pair $i =< ID, type >$, where ID is an unique identifier for each item and $type$ represents the class of the item (e.g. normative set).
- A finite set of discrete values that represent the feedback that an agent offers to the dialogue participants about their recommendation, denoted *Usefulness*, with elements *-1: Inappropriate, 0: Useless or 1: Useful.*
- A finite set of variables that represent the agent's preferences when it asks for a recommendation, denoted *Preferences*, with elements $P = \{p_1, ..., p_q\}$.
- A function *preference:* $P \rightarrow 2^{F \times V}$, which maps each Preference of an agent to a set of pairs $< f, v >$, where f ∈ Features and v ∈ Values.
- A function *feedback:* $Item \rightarrow Usefulness$, which maps an Item that an agent has recommended to one of the possible values of usefulness.

Finally, the interaction protocol between the agents of the network has been modelled as a formal dialogue game with the components identified by McBurney and Parsons [11]:

- Commencement Rules: The process consists in a set of parallel dialogues between the initiator and the NAs (who do not speak directly between them). In each step of the dialogue, either the initiator makes a move or the NA answers it by posing the permissible locutions following the dialogue rules. Note that, although the information provided by a NA is not directly accessible by another, the initiator acts as a mediator and is able to use this information when speaking with other NAs. The dialogue starts when an agent asks the THOMAS SF component for a set of NAs. Then, the initiator of the dialogue uses its list of friends to select a subset of those NAs and send them a request for a normative set recommendation. When the NAs have received the recommendation request with the preferences of the initiator, they use their own reasoning mechanisms to match these preferences with their knowledge database and offer a recommendation.

Eventually, NAs can decide not to engage in the recommendation dialogue.

- Locutions:
  - Statements: they are the permissible locutions in the dialogue and their compounds *(propose(Item, Preferences)*, *accept*, *reject*, *noCommitment(Item)* and *assert(Data))*.
  - Questions: *propose(Item,Preferences)?* is used to request recommendations from friends. Also, the locution *assert({Preferences})?* asks the initiator for more information about its preferences when there are some unspecified. Agents are not committed to answering questions.
  - Critical Attacks (CA): the locution *pose(CA)* poses a critical attack associated with one of the critical questions of the Argument from Expert Opinion scheme. In our model we assume (a) that all NAs have some knowledge about the items of the domain and hence, every NA can be considered expert to some extent (*Field question*); (b) that NAs are rational and always propose the recommendation that, using their reasoning mechanisms, better fits the preferences of the initiator (*Opinion question*) and (c) that NAs are honest and their recommendations and arguments are based on their own knowledge (*Backup evidence question*). Thus, permissible attacks to recommendations are: (i) questioning the degree of expertise of the proponent (*Expertise question*); (ii) demonstrating that the proponent is not personally reliable as recommender (*Trustworthiness question*); or (iii) demonstrating that the proponent's recommendation is not consistent with the one of other expert with equal or greater degree of expertise (*Consistency question*). The burden of proof in the case of the Trustworthiness and Consistency attacks falls on the initiator and therefore, if the attack is challenged, it is committed to providing the proponent with arguments that justify those criticisms.
  - Challenges: *why(locution)?*, where locution can be a recommendation proposal or a critical attack, requests arguments that support them.
- Commitment Rules:
  - Before the assertion of the locution *propose(Item, Preferences)*, if proponents have different recommendations that match the preferences of the initiator, they are committed to ask it for specifying those preferences that could stand out a recommendation from the rest.
  - The assertion of the locution *propose(Item, Preferences)* commits the proponent to the assumptions that the game makes about the critical questions of the Argument from Expert Opinion scheme.
  - The assertion of the locution *noCommitment(Item)*, withdraws the recommendation made by the proponent and frees it from all commitments.
  - The assertion of the locution *why(propose(Item, Preferences))?* commits the proponent of the recommendation either to providing the initiator with a justification about its proposal or to withdrawing it.
  - If a critical attack *pose(CA)* is challenged, the initiator is committed to providing arguments to justify it.
- Dialogue Rules:
  1) The initiator's request opens the dialogue and each NA can answer it by a request for more information about the properties that characterise the problem, a normative set proposal or a rejection to provide the assistance service.
  2) NAs can ask for more information while the initiator accedes to provide it. Finally, the NA makes a proposal (*proponent* role) or rejects to provide the service.
  3) Each normative set proposal can be followed by a request for an argument supporting the recommendation, or by its acceptance or rejection.
  4) If a recommendation is challenged, the proponent must show its argument for recommending such normative set or withdraw the recommendation.
  5) The initiator can reply to the argument of a NA by accepting the recommendation, by rejecting it or by posing a critical attack associated with the *Argument from Expert Opinion* explained in section II-B. Possible attacks in our context are: questioning the degree of expertise of the NA or demonstrating that the recommendation of the NA is not consistent with other NA recommendations with equal or greater degree of confidence or expertise.
  6) Trustworthiness and Consistency attacks can also be challenged by the NA. Then, the initiator must provide an argument supporting the attack.
  7) NAs can rebut attacks or else, withdraw their recommendations.
  8) Finally, the initiator can accept the argument of a NA and choose its recommendation, preliminary accept its argument but pose another attack or reject the argument and hence, the recommendation (ending the dialogue with this NA).
- Termination Rules: The initiator agent keeps at every moment the entire control of the recommendation process. The preferences of the initiator do not change during the dialogue and proponents are assumed to be rational and honest. Thus, once a recommendation has been proposed, it cannot be changed. Proponents are only allowed to propose items, to answer challenges and critical attacks and to withdraw their recommendations. The game ends when the initiator has taken a decision among the set of available recommendations, which can happen at any time during the dialogue. Afterwards, the proponents are informed and receive the *accept/reject* locutions to their recommendations. However, after a maximum time is exceeded, the recommendation partially accepted to this step of the dialogue is taken as the final decision.

C. Decision-Making Process of NAs

The basic decision policy that follows every NA is to do its best to convince the initiator that its normative set recommendation is the most appropriate one to solve the coordination problem. There are two different types of arguments that NAs can use to persuade the initiator: arguments for justifying a recommendation and arguments for rebutting an attack. An argument of the former type consists in a set of common attributes among the problem characterisation and the proposed solution. Regarding the rebutting arguments, they consists on a partial ordering relation between confidence or expertise degrees. The simplest case is to rebut an *expertise attack*, since the NA can only show its expertise degree (note that expertise degrees are private and this attack is thought to provide the initiator with this information).

In the case of a *trustworthiness attack*, the initiator i attacks the recommendation of the proponent p because it has received a different recommendation from other NA n with a higher confidence degree for the initiator. Then, p can rebut the attack if the conditions expressed on equation 3 hold:

$$\begin{gathered} Argument\ AR = (c, <, c) \\ partial\ ordering\ relation\ \delta = \{<\} \\ attacks \subseteq AR \times AR \\ Case(a) \\ (c_{np}, <, c_{in})\ attacks\ (c_{ip}, <, c_{in})\ iff \\ c_{pn} < c_{ip} \\ Case(b) \\ (c_{in}, <, c_{pk})\ attacks\ (c_{ip}, <, c_{in})\ iff \\ 0 < c_{ip} < c_{in} < c_{pk} \end{gathered} \tag{3}$$

In case (a), p can rebut the attack if it personally knows n and its confidence degree in this NA c_{pn} is lower than the confidence degree c_{ip} of the initiator in p. In case (b), p can rebut the attack if the recommendation was propagated to the NA k and the confidence degree c_{pk} of p in this NA is higher than the confidence degree c_{ip} that the initiator has in p and also the confidence degree c_{in} that the initiator has in the NA n.

In the case of a *consistency attack*, the initiator i attacks the recommendation of the proponent p because it has received a different recommendation from other NA n with a higher expertise degree for the initiator. Then, p can rebut the attack if the conditions expressed on equation 4 hold:

$$\begin{gathered} Argument\ AR = (e, <, e) \\ partial\ ordering\ relation\ \delta = \{<\} \\ attacks \subseteq AR \times AR \\ (e_n, <, e_k)\ attacks\ (e_p, <, e_n)\ iff \\ e_p < e_n < e_k \end{gathered} \tag{4}$$

In this case, p can rebut the attack if the recommendation was propagated to the NA k and the expertise degree e_k of this NA k is higher than the expertise degree e_p of p and also the expertise degree e_n of the NA n.

IV. Normative Context Definition

In order to show the operation of our dialogue game protocol, we have applied it for solving a coordination problem. More concretely, the addressed problem consists in selecting the most suitable norms for a new organization that an agent wants to create.

A. Problem Formalization

In our approach, coordination is achieved by means of norms that regulate the activities of agents. Therefore, a coordination problem is formalised as a structure of the type γ where $\gamma = \{\gamma_0, \gamma_1, ..., \gamma_k\}$ is a set of attributes or properties that characterise the problem; $\gamma_i \in D_i$, being D_i the domain associated to the property i.

In the application example, the addressed problem consists on the definition of the normative context for a new VO according to its desirable features. This problem is a particular case of a coordination problem. Thus, a problem of normative context definition is defined as γ where $\gamma = \{s, m, v, c, f\}$ and

- $s =< who, how >$ is a property that determines who can change the structural components of the system (i.e. roles, units and norms); where $who \in \{none, supervisor, all\}$ and $how \in \{increase, decrease, modify\}$.
- $m \in \mathbb{N}$ is the property that specifies the cardinality of the VO members.
- $v \in \{none, supervisor, members, all\}$ is the visibility property that establishes the access rights to the information of the VO.
- $c \in \{none, supervisor, all\}$ is the property that determines if the expulsion service can be used as a control mechanism.
- $f =< who, how >$ is the property that defines who and how the functionalities, i.e. services, of the VO can be changed; where $who \in \{none, supervisor, all\}$ and $how \in \{increase, decrease, modify\}$.

Thus, items in this domain correspond to normative sets (i.e. $< ID, normativeSet >$) and the preferences of the initiator correspond to the properties that characterise the coordination problem (i.e. γ).

B. Application Example

The social network of normative assistants can be implemented in THOMAS as a VO (named *NormativeAssistance*). It is a group of agents formed by all agents that can provide normative assistance (*Assistants*) to other agents. Therefore, agents in charge of the maintenance of the normative context that regulates a specific group can request an advice for norms effective in some scenario. All the communication and functionalities specified by the protocol are carried out as services.

As an example, Figure 2 contains an overview of the services performed by a set of agents which are playing the proposed dialogue game. Let us consider the case that an agent (*initiator*) has requested the service for creating a new VO in THOMAS. In this situation, the agent needs to define the normative context that will regulate its VO. However, this agent does not know a priory the suitable set of norms for its VO. Thus, it can ask for a *NormativeAssistance* service to

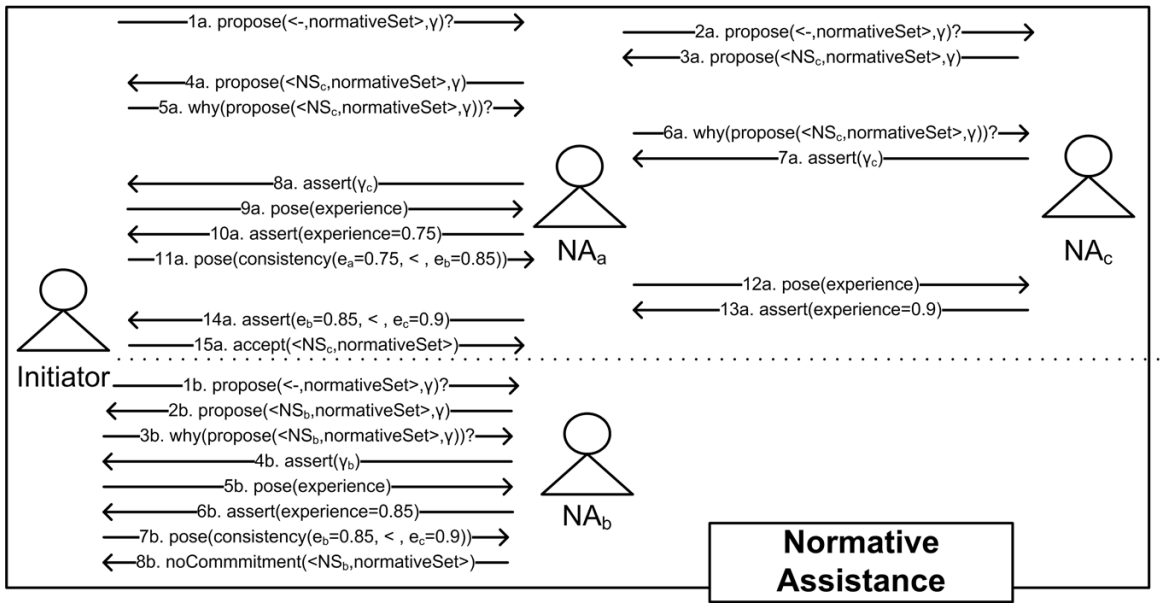

Fig. 2. Example of an execution of the proposed dialogue game

the SF, who will apply its techniques for service searching and composition to discover this recommendation service. If it exists, the initiator agent will obtain the list of service providers from the SF.

As previously mentioned, the *initiator* selects the NAs from the provider list according to its previous experiences. In this example, the *initiator* requests the *NormativeAssistance* service to NA_a and NA_b (Figure 2 steps 1a and 1b). This service request implicitly utters the *propose* locution which starts the dialogue game. Therefore, the *initiator* carries out two different dialogues concurrently (labelled as a and b). The NA_a does not have the requested information in its norm database and thus, it propagates the request to its friend NA_c (step 2a). Thus, the set of players in this example is defined as $Agents = \{initiator, NA_a, NA_b, NA_c\}$. NA_c queries its norm database and proposes the normative set NS_c (step 3a and its propagation in step 4a). Moreover, the NA_b proposes NS_b as a solution to the coordination problem characterized by γ (step 2b). Since the *initiator* does not have enough information for choosing between NS_c and NS_b, it requests a justification of the recommended normative sets (steps 5a, its propagation 6a and step 3b). After receiving the justification of the proposed recommendation (steps 7a, its propagation 8a and step 4b) the *initiator* still does not have enough information for taking a decision. Thus, it poses an expertise attack (steps 9a and 5b) to NA_a and NA_b, which answer by informing about their expertise degree in steps 10a and 6b, respectively. Since the expertise of NA_b is higher than NA_a, the initiator poses a consistency attack to NA_a (step 11a). As a consequence, NA_a poses an expertise attack to NA_c (step 12a) to know its expertise degree (provided in step 13a). Hence, NA_a is able to rebut the confidence attack by showing that the expertise level of NS_c is higher than NS_b (step 14a). Then, the initiator poses a consistency attack to NA_b which cannot rebut it and hence, withdraws its recommendation (steps 7b and 8b, respectively). Finally, the *initiator* accepts the normative set NS_c as the best solution for the current coordination problem (step 15a).

After this recommendation dialogue, the *initiator* applies the normative set NS_c in its VO and evaluates its effectiveness. The definition of the recommendation evaluation process carried out by agents is out of the scope of this paper. Therefore, the initiator is able to update its confidence degrees of the agents that engaged in the dialogue (NA_a, NA_b and NA_c). This last agent is added to its friend list. In addition, the *initiator* sends feedback information about the utility of the recommendation to all NAs, which update their expertise degrees.

V. Conclusion

This paper addresses the problem consisting in the dynamical definition of norms for agent societies. In open societies, which are situated in a dynamic environment and are formed by heterogeneous autonomous agents, the existence of mechanisms for adapting and modifying norms is essential. Therefore, our approach proposes the employment of dialogue games to model the interactions produced between agents that must reach an agreement on the use of norms. This work takes as basis well known proposals on dialogue games. In addition, the THOMAS architecture has been used as an infrastructure for the dialogues among agents.

We have adapted the ASD dialogue game to fit our objective of supporting norm emergence in an open and social context. Finally, an application example has been presented in order to show both the performance of the dialogue game protocol and its usefulness as a mechanism for managing the solving process of a coordination problem through norms. As future work we plan to apply this approach to more complex scenarios in which a more elaborated process is required for solving these coordination problems.

ACKNOWLEDGMENT

This work was partially supported by CONSOLIDER-INGENIO 2010 under grant CSD2007-00022, by the Spanish government and GVA funds under TIN2006-14630-C0301 and PROMETEO/2008/051 projects and by the FPU grant AP-2007-01256 awarded to N. Criado.

REFERENCES

[1] Argente, E., Criado, N., Julián, V., Botti, V.: Norms for Agent Service Controlling. EUMAS-08, pp. 1-15, (2008).

[2] Artikis, A., Pitt, J. :A formal model of open agent societies. In Proceedings of the Fifth international Conference on Autonomous Agents (AGENTS). ACM, New York, NY, 192-193, (2001).

[3] Boella, G., Hulstijn, J., van der Torre, L.: Virtual organizations as normative multiagent systems. HICSS IEEE Computer Society, (2005).

[4] Carrascosa, C., Giret, A., Julián, V., Rebollo, M., Argente, E., Botti, V.: Service Oriented Multi-agent Systems: An open architecture. AAMAS-09, pp. 1291-1292, (2009).

[5] Giret, A., Julian, V., Rebollo, M., Argente, E., Carrascosa C., Botti,V.:An Open Architecture for Service-Oriented Virtual Organizations. Seventh international Workshop on Programming Multi-Agent Systems(PROMAS), pp. 74-88, (2009).

[6] Heras, S., Navarro, M., Botti ,V., Julián, V.: Applying Dialogue Games to Manage Recommendation in Social Networks. ArgMAS-09, pp. 55-70, (2009).

[7] López y López, F., Luck, M., d'Inverno, M.: Constraining autonomy through norms. AAMAS-02, pp. 674-681, (2002).

[8] Luck, M., McBurney, P., Shehory, O. and Willmott, S.:Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing). University of Southampton, (2005).

[9] Malone, T., Crowston, K.: The Interdisciplinary Study of Coordination. ACM Computing Surveys, 1994 (March), vol. 26, no. 1, pp. 87-119, (1994).

[10] McBurney, P., Parsons, S.: Dialogue Games in Multi-Agent Systems. Informal Logic. Special Issue on Applications of Argumentation in Computer Science. Vol. 22, no. 3, pp. 257-274, (2002).

[11] McBurney, P., Parsons, S.: Games that Agents Play: A Formal Framework for dialogues between Autonomous Agents. Journal of Logic, Language and Information, vol. 12, no. 2, pp. 315-334, (2002).

[12] O'Donovan, J., Smyth, B.: Trust in recommender systems. 10th International Conference on Intelligent User Interfaces, pp. 167-174, (2005).

[13] Reed, C., Walton. D.: Argumentation Schemes in Dialogue. In Dissensus and the Search for Common Ground, OSSA-07, volume CD-ROM, pp. 1-11, (2007).

[14] Schelling, T.: The Strategy of Conflict. Harvard University Press, Cambridge, (1960).

[15] Shoham, Y., Tennenholtz, M.: On social laws for artificial agent societies: off-line design. Artificial Intelligence, v.73 n.1-2, p.231-252, (1995).

[16] Sen, S., Airiau, S.: Emergence of norms through social learning. In Proceedings of the International Joint Conference on Artificial Intelligence(IJCAI), pp. 1507–1512, (2007).

[17] Verhagen, H.: Norm Autonomous Agents. PhD thesis, Dept. Computer Science, Stockholm University, (2000).

[18] Walton, D.: Appeal to Expert Opinion. Penn State Press, (1997).

Guidelines to apply CBR in Real-Time Multi-Agent Systems

Martí Navarro and Stella Heras and Vicente Julián

***Abstract*—In real-time Multi-Agent Systems, Real-Time Agents merge intelligent deliberative techniques with real-time reactive actions in a distributed environment. CBR has been successfully applied in Multi-Agent Systems as deliberative mechanism for agents. However, in the case of Real-Time Multi-Agent Systems the temporal restrictions of their Real-Time Agents make their deliberation process to be temporally bounded. Therefore, this paper presents a guide to temporally bound the CBR to adapt it to be used as deliberative mechanism for Real-Time Agents.**

***Index Terms*—Real-Time Multi-Agent Systems, Case-Based Reasoning.**

I. Introduction

The need for developing software solutions applied to complex, non-dynamic and frequently, non-completely specified environments, has contributed to the confluence of two important research areas, the Artificial Intelligence (AI) and the Real-Time System (RTS). Inside the Artificial Intelligence framework, Multi-Agent Systems paradigm (MAS) represents an appropriate approach for solving inherently distributed problems. The work presented in this paper is planned over the existent relationship between MAS and RTS. This work covers the problem of Real-Time Agents (RTAs), which merge intelligent deliberative techniques with real-time reactive actions in a distributed environment.

A Real-Time Artificial Intelligence System (RTAIS) is a system that must accomplish critical processes under a dynamic environment with temporal restrictions by using AI techniques. Here, anytime algorithms [4] and approximate processing [7] are the most promising algorithms. One line of research in RTAI has been to build applications or architectures that embody real-time concerns in many components[7], such as Guardian [9], Phoenix [10], CIRCA/ SA-CIRCA [8], [17] and ARTIS [6], [2]. An appropriated agent for real-time environments must accomplish its goals, responsibilities and tasks with the added difficulty of temporal restrictions. Thus, an RTA can be defined as an agent with temporal restrictions in, at least, one of its responsibilities. The RTA may have its interactions bounded, and this modification will affect all the communication processes in the MAS where the RTA is located.

The main problem in the architecture of an RTA concerns the deliberation process. This process commonly uses AI techniques as problem-solving methods to compute more intelligent actions. However, the temporal restrictions of RTAs give rise to the necessity of providing techniques that allow their response times to be bounded. These techniques are based on RTAIS techniques [7]. In addition, in an RTA an efficient integration of high-level, deliberative planning processes within reactive processes is necessary. These complex deliberative processes, which allow the agent to reason, to adapt and learn, are unbounded and it is difficult to integrate them in real-time systems. However, their main drawback lies in finding a mechanism that permits their efficient and temporal bounded execution.

In view of the successful applications reported in the literature (see Section III), we propose a model where RTAs use a CBR method as deliberative mechanism to take decisions. However, the execution of this CBR method must be bounded in order to observe the RTA temporal restrictions. Thus, this paper presents a guide to temporally bound the CBR cycle to adapt it to be used as a deliberative mechanism for RTAs. The paper is structure as follow: Section II introduces the concept of Real-Time Agent; Section III reviews related successful applications of the CBR method in MAS; Section IV presents a guide with the principal facts to take into account to temporally bound the CBR cycle and finally, some conclusions are shown in Section V.

II. Real-Time Agent

A Real-Time Agent (RTA) is an agent composed of a series of tasks, some of which have temporal constraints [11]. In these agents, it is also necessary to take into account the temporal correctness, which is expressed by means of a set of temporal restrictions that are imposed by the environment. The RTA must, therefore, ensure the fulfilment of such temporal restrictions. By extension, a Real-Time Multi-Agent System (RTMAS) is a multi-agent system with at least one RTA [11]. Systems of this type require the inclusion of temporal representation in the communication process, management of a unique global time, and the use of real-time communication [23].

It is well-known that a typical real-time system is made up of a set of tasks characterized by a deadline, a period, a worst-case execution time and an assigned priority. These restrictions in the system functionality affect the features of an agent that needs to be modelled as a real-time system. The main problem is that if its tasks are not temporally bounded properly, it is not possible to guarantee the fulfilment of the tasks before a deadline is expired and to schedule a plan with these tasks.

The reasoning process of the RTA must be temporally bounded to allow it to perform the tasks for deciding the

Martí Navarro is with Polytechnic University of Valencia. E-mail: mavarro@dsic.upv.es

Stella Heras is with University of Polytechnic University of Valencia. E-mail: sheras@dsic.upv.es

Vicente Julián is with University of Polytechnic University of Valencia. E-mail: vinglada@dsic.upv.es

strategy to reach its objectives. In this way, the RTA will be able to determine whether it has enough time to deliberate and to take into account the temporal cost of its cognitive task when it plans the execution of new tasks. Next sections review CBR applications to MAS and propose a temporally bounded CBR method as deliberative mechanism for the cognitive task of the agent.

III. CBR AS DELIBERATIVE MECHANISM FOR AGENTS

In the AI research, the combination of several AI techniques to cope with specific functionalities in hybrid systems has a long history of successful applications. A CBR system provides agent-based systems with the ability to reason and learn autonomously from the experience of agents. These systems propose solutions for solving a current problem by reusing or adapting other solutions that were applied in similar previous problems. With this aim, the system has a case-base that stores its knowledge about past problems together with the solution applied in each case. The most common architecture of a CBR system consists of four phases [1]: the first one is the *Retrieval* phase, where the most similar cases are retrieved from the case-base; then, in the *Reuse* phase, those cases are reused to try to solve the new problem at hand; after that, in the *Revise* phase the solution achieved is revised and adapted to fit the current problem and; finally, in the *Retain* phase, the new case is stored in the case-base and hence, the system learns from new experiences. The integration of CBR systems and MAS has been studied following many approaches. Therefore, the literature of this area reports research on systems that integrate a CBR engine as a part of the system itself [12], other MAS that provide some or each of their agents with CBR capabilities or even, the development of BDI agents following a CBR methodology [3]. This section is focused on the review of the second approach, CBR applied to MAS, since it fits the scope of our paper.

Since the 90's, the synergies between MAS and CBR are many, although the approaches differ. One early approach was the development of multi-agent CBR systems, which are MAS with cooperative agents characterized by the distribution of their case-bases and/or certain phases of the CBR cycle between them [13], [16], [20]. The main effort in this research area is focused on the policies that agents follow to manage the CBR cycle.

The application of CBR to manage argumentation in MAS is a different and more recent approach that has produced important contributions both in the areas of AI and argumentation theory. In this field, important works are the case-based negotiation model for reflective agents proposed in [22], the new case-based selection model ProCLAIM [24], which extended the architecture of the decision support MAS for the organ donation CARREL+ and the Argumentation Based Multi-Agent Learning (AMAL) framework [19], which features a set of agents that try to solve a classification problem by aggregating their expert knowledge.

Furthermore, an area where the integration between CBR and agent techniques has produced a huge amount of successful applications is the robot navigation domain. An important contribution was a case-based model for managing the ROBOCATS system [15], playing in the Robocup league. Also, the RUPART system [5], which features a hybrid planner for a mobile robot that delivers mail in real-time. Another application of CBR to manage autonomous navigation tasks was a system for the automatic selection and modification of assemblage parameters proposed in [14]. Finally, an important research that applies CBR to MAS with mobile robots modelled team playing behaviour in the robot soccer domain by using CBR [21].

The cited above are outstanding examples of systems that join research efforts and results of both CBR and MAS, but they are only a sample reported in the literature of this prolific area. However, few of these systems cope with the problem of applying CBR as deliberative engine for agents in MAS with real-time constraints. In this case, the case-based reasoning cycle must observe temporal restrictions. The next section tackles this challenge and provides solutions to deal with it.

IV. TEMPORALLY-BOUNDED CBR

CBR systems are highly dependent on their application domain. Therefore, designing a general CBR model that might be suitable for any type of real-time domain (hard or soft) is, to date, unattainable. In real-time environments, the CBR phases must be temporally bounded to ensure that solutions are produced on time. In this section, we present some guidelines with the minimum requirements to be taken into account to implement a CBR method in real-time environments.

The design decision about the data structure of the case-base and the different algorithms that implement each phase are important factors to determine the execution time of the CBR cycle. The number of cases in the case-base is another parameter that affects the temporal cost of the retrieval and retain phases. Thus, a maximum number of cases must be defined by the designer. Note that, usually, the temporal cost of the algorithms that implement these phases depend on this number.

For instance, let us assume that the designer chooses a hash table as data structure for the case-base. This table is a data structure that associates keys to concrete values. Search is the main operation that it supports in an efficient way: it allows the access to elements (e.g. phone and address) by using a hash function to transform a generated key (e.g. owner name or account) to a hash number that is used to locate the desired value. The average time to make searches in hash tables is constant and defined as $O(n)$ in the worst case. Therefore, if the cases are stored as entries in a hash table, the maximum time to look for a case depends on the number of cases in the table (i.e. $O(number\_cases)$). Similarly, if the case-base is structured as an auto-balanced binary tree the search time in the case-base in the worst case would be $O(log(n))$.

In this research, we propose a modification of the classic CBR cycle (Figure 1) to adapt it to be applied in real-time domains. Figure 2 shows a graphical representation of our approach. First, we group the four reasoning phases that implement the cognitive task of the real-time agent in two stages: the learning stage, which consists of the revise and

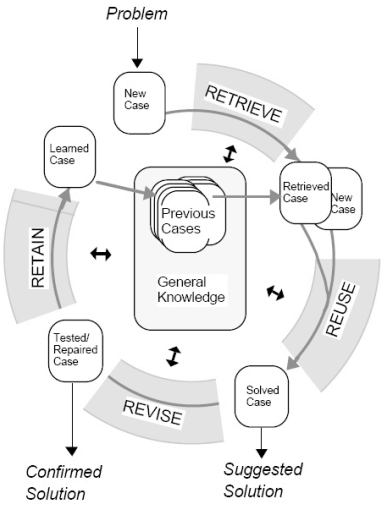

Fig. 1. Classic CBR cycle.

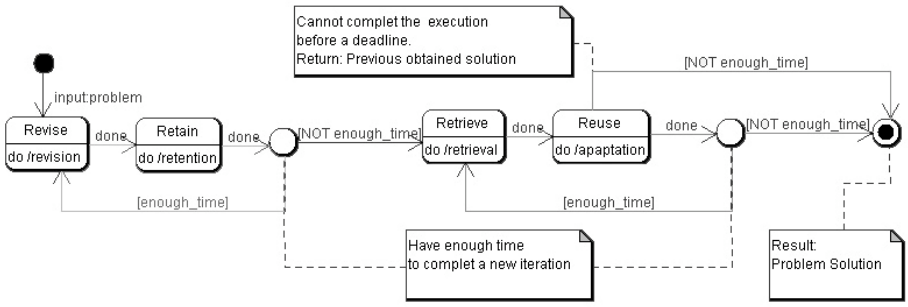

Fig. 2. Temporally Bounded CBR cycle.

retain phases and the deliberative stage, which includes the retrieve and reuse phases. Both phases will have scheduled their own execution times. In this way, the designer can choose between assigning more time to the deliberative stage (and hence, to design more 'intelligent' agents) or else, keeping more time for the learning stage (and thus, to design agents that are more sensible to updates).

Following, the operation of our Time Bounded CBR cycle (TB-CBR) is explained. Firstly, the main difference that can be observed between the classical CBR cycle and the TB-CBR cycle is the starting phase. Our real-time application domain and the restricted size of the case-base gives rise to the need of keeping the case-base as updated as possible. Commonly, recent changes in the case-base will affect the potential solution that the CBR cycle is able to provide for a current problem. Therefore, the TB-CBR cycle starts by the learning stage, checking if there are previous cases waiting for being revised and possible stored in the case-base. In our model, the solutions provided at the end of the deliberative stage will be stored in a solution list while a feedback about their utility is received. When each new CBR cycle begins, this list is accessed and while there is enough time, the learning stage of those cases whose solution feedback has been recently received is executed. In case the list is empty, this process is omitted.

After that, the deliberative stage is executed. Thus, the retrieval algorithm is used to search the case-base and retrieve a case that is similar to the current case (i.e. the one that characterises the problem to solve). Each time a similar case is found, it is sent to the reuse phase where it is transformed to a suitable solution for the current problem by using a reuse algorithm. Therefore, at the end of each iteration of the deliberative stage, the TB-CBR method is able to provide a solution for the problem at hand, although this solution can be improved in next iterations as long as the deliberative stage has enough time to perform them.

Hence, the temporal cost of executing the cognitive task is greater than or equal to the sum of the execution times of the learning and deliberative stages (as shown in equation 1):

$$\begin{aligned} t_{cognitiveTask} &\geq t_{learning} + t_{deliberative} \\ t_{learning} &\geq (t_{revise} + t_{retain}) * n \\ t_{deliberative} &\geq (t_{retrieve} + t_{reuse}) * m \end{aligned} \quad (1)$$

where $t_{learning}$ and $t_{deliberative}$ are the total execution time of the learning and deliberative stages; t_x is the execution time of the phase x and *n* and *m* are the number of iterations of the learning and deliberative stages respectively. The requirements

needed to temporally bound each phase of the TB-CBR cycle are explained below. In order to bound the temporal cost of the algorithms that implement these phases and to ensure an adequate temporal control of them, the execution time of these algorithms is approximated by its *worst-case execution time* (WCET). The WCET sets a maximum threshold for the temporal execution of each algorithm and thus, this prevents the temporal constraints of the system to be broken.

A. Revise Phase

In order to keep the case-base as up to date as possible, the revise phase is performed first. During this phase, the accuracy of the final solutions obtained in previous executions of the TB-CBR cycle is checked. The revision algorithm (i.e. $f_{revision}$) only checks one solution per iteration, fixing the potential problems that it had in case of erroneous results. The outcome of this phase is used to update the case-base. Thus, the maximum temporal cost of this phase is bounded by the WCET of the revision algorithm:

$$t_{revise} = WCET(f_{revision}(solution)) \tag{2}$$

Note that, in order to guarantee a known maximum execution time, this checking must be performed automatically by the computer without human interference. This WCET does not depend on the number of stored solutions or the number of cases in the case-base and again, must be determined by the designer of the algorithm.

B. Retain phase

In this phase it is decided whether a checked solution must be added as a new case in the case-base. Here, keeping the maximum size of the case-base is crucial, since the temporal cost of most retention algorithms (i.e. $f_{retention}$) depends on this size. If there is a case in the case-base that is similar enough to the current case, this case (its problem description and solution) is updated if necessary. On the contrary, if there is not a case that represents the problem solved, a new case is created and added to the case-base. In order to keep the maximum size of the case-base, this could entail removing an old case from it. This decision should be taken by the retention algorithm. Nevertheless, the maximum temporal cost that the retain phase needs to execute one iteration is the retention algorithm WCET.

$$t_{retain} = WCET(f_{retention}(solution, CaseBase)) \tag{3}$$

C. Retrieve Phase

In the retrieve phase, the retrieval algorithm (i.e. $f_{retrieval}$) is executed to find a case that is similar to the current problem (i.e. $currentCase$) in the case-base. Since WCET depends on the structure of the case-base and its number of cases, the designer must calculate this WCET and use this time to estimate the necessary time to execute an iteration of the retrieval algorithm.

$$t_{retrieve} = WCET(f_{retrieval}(currentCase, CaseBase)) \tag{4}$$

Each execution of the retrieval algorithm will provide a unique case similar to the current problem (if it exists in the case-base). This result is used as input for the reuse phase. However, in next iterations of the deliberative stage more similar cases can be retrieved with the intention to provide a more accurate solution for the problem.

D. Reuse Phase

In this phase, the cases obtained from the retrieve phase are adapted to use them as a potential solution for the current problem. These cases are stored in a list of selected cases (i.e. $cases_{list}$). Each time the reuse phase is launched, the adaptation algorithm (i.e. $f_{adaptation}$) searches this list and produces a solution by adapting a single case or a set of cases to fit the context of the current problem to solve. Therefore, the execution time of this algorithm depends on the number of cases that the algorithm is working with.

$$t_{reuse} = \begin{cases} WCET(f_{adaptation}(firstCase)) \\ f_{adaptation}(listOfCases) \end{cases} \tag{5}$$

To guarantee that the RTA assigns enough time to perform the cognitive task and provides at least one solution, the designer must know the WCET to execute one iteration of the adaptation algorithm (i.e. $f_{adaptation}(firstCase)$). In order to control the execution time of the adaptation algorithm in subsequent iterations (i.e. $f_{adaptation}(listOfCases)$), the RTA must be able to stop the execution of the algorithm in case that it realises that the assigned time to complete the deliberative stage will be overcome. Then, the RTA provides the best solution among the solutions completed in previous iterations. This solution is stored in a list of solutions for being verified in the learning stage.

V. Conclusion

The main contribution of this article is to set some guidelines to develop a CBR method for real-time systems. In this type of systems, timing requirements must be previously known in order to guarantee the correctness of the system. However, since the execution time of each CBR method depends on the specific algorithms that have been used to implement the CBR cycle, a general estimation cannot be made. Therefore, this work provides the designer of the system with a new approach for the CBR cycle, called TB-CBR, which eases the process of bounding each phase of the CBR method. This method implements the CBR cycle in two stages: the deliberative stage, whose execution is mandatory; and the learning stage, whose execution can be optional depending on the temporal requirements.

This TB-CBR proposal has been implemented and tested in an example that consists in a system that manages the mail in a department plant by using mobile robots [18]. In this example, robots deliberate to know whether they have enough time to deliver mail. This deliberation is implemented by using a TB-CBR method.

ACKNOWLEDGMENT

This work was partially supported by CONSOLIDER-INGENIO 2010 under grant CSD2007-00022 and by the Spanish government and GVA funds under TIN2006- 14630-C0301 and PROMETEO/2008/051 projects.

REFERENCES

[1] A. Aamodt and E. Plaza, *Case-based reasoning; Foundational issues, methodological variations, and system approaches*. AI Communications, vol. 7, no. 1, pp. 39-59, 1994.

[2] C. Carrascosa and A. Terrasa and A. García-Fornes and A. Espinosa and V. Botti, *A Meta-Reasoning Model for Hard Real-Time Agents*. CAEPIA, vol. 4177, pp. 42-51, 2005.

[3] J. M. Corchado and A. Pellicer, *Development of CBR-BDI Agents*. International Journal of Computer Science and Applications, vol. 2, no. 1, pp. 25-32, 2005.

[4] T. Dean and M. Boddy, *An analysis of time-dependent planning*. Proc. of the 7th National Conference on Artificial Intelligence, pp. 49-54, 1988.

[5] S. E. Fox and P. Anderson-Sprecher, *Robot Navigation: Using Integrated Retrieval of Behaviors and Routes*. Proc. of FLAIRS Conference, pp. 346-351, 2006.

[6] A. García-Fornes, *ARTIS: Un modelo y una arquitectura para sistemas de tiempo real inteligentes*. Ph.D. Dissertation, DSIC, U. Politecnica Valencia, 1996.

[7] A. Garvey and V. Lesser, *A survey of research in deliberative Real-Time Artificial Intelligence*. The Journal of Real-Time Systems, vol. 6, pp. 317-347, 1994.

[8] R. P. Goldman and D. J. Musliner and K. D Krebsbach, *Managing Online Self-Adaptation in Real-Time Environments*. Proc. of 2nd Int. Workshop on Self Adaptive Software, 2001.

[9] B. Hayes-Roth and R. Washington and D. Ash and A. Collinot and A. Vina and A. Seiver, *Guardian: A prototype intensive-care monitoring agent*. Artificial Intelligence in Medicine, vol. 4, pp. 165-185, 1992.

[10] A. E. Howe and D. M. Hart and P. R. Cohen, *Addressing real-time constraints in the design of autonomous agents*. The Journal of Real-Time Systems, vol. 2, pp. 81-97, 1990.

[11] V. J. Julián and V. Botti, *Developing real-time multi-agent systems*. ICAE, vol. 11, no. 2, pp. 150-165, 2004.

[12] N. Karacapilidis and D. Papadias, *Computer supported argumentation and collaborative decision-making: the HERMES system*. Information Systems, vol. 26, no. 4, pp. 259-77, 2001.

[13] D. B. Leake and R. Sooriamurthi, *When Two Case Bases Are Better Than One: Exploiting Multiple Case Bases*. Proceedings of ICCBR, 2001.

[14] M. Likhachev and M. Kaess and R. C. Arkin, *Learning Behavioral Parameterization Using Spatio-Temporal Case-Based Reasoning*. Proc. of ICRA, vol. 2, pp. 1282-1289, 2002.

[15] C. Marling and M. Tomko and M. Gillen and D. Alexander and D. Chelberg, *Case-based reasoning for planning and world modeling in the robocup small size league*. Workshop on Issues in Designing Physical Agents for Dynamic Real-Time Environments, IJCAI, 2003.

[16] L. Mc Ginty and B. Smyth, *Collaborative Case-Based Reasoning: Applications in Personalised Route Planning*. Proceedings of ICCBR, pp. 362-376, 2001.

[17] D. J. Musliner and J. A Hendler and A. K. Agrawala and E. H. Durfee and J. K. Strosnider and C. J. Paul, *The Challenge of Real-Time in AI*. IEEE Computer, January, pp. 58-66, 1995.

[18] M. Navarro and S. Heras and V. Julián, *Ensuring Time in Real-Time Commitments*. In Proc. of IBERAMIA, pp. 183-192, 2008.

[19] S. Ontañon and E. Plaza, *Learning and Joint Deliberation through Argumentation in Multi-Agent Systems*. Proc. of AAMAS, 2007.

[20] E. Plaza and J. L. Arcos and F. Martn, *Cooperative Case-Based Reasoning*. LNAI, vol. 1221, pp.180-201, 1997.

[21] R. Ros and R. López de Mántaras and J. L. Arcos and M. Veloso, *Team Playing Behavior in Robot Soccer: A Case-Based Approach*. Proc. of ICCBR, vol. 4626, pp. 46-60, 2007.

[22] L. K. Soh and C. Tsatsoulis, *A Real-Time Negotiation Model and a Multi-Agent Sensor Network Implementation*. AAMAS, vol. 11, no. 3, pp. 215-271, 2005.

[23] J. Soler and V. Julián and A. García-Fornes and V. Botti, *Real-Time Extensions in Multi-agent Communication*. LNAI 3040, pp. 468-477, 2003.

[24] P. Tolchinsky and S. Modgil and U. Cortés and M. Sánchez-Marré, *CBR and Argument Schemes for Collaborative Decision Making*. Proc. COMMA, vol. 144, pp. 71-82, 2006.