INTERNATIONAL JOURNAL OF INTELLIGENT CONTROL AND SYSTEMS
VOL.12, NO.2 JUNE 2007 179-185

Role Suggestion for Agents by Overhearing

Giacomo CABRI, Luca FERRARI, Letizia LEONARDI and Raffaele QUITADAMO

Abstract- **Software agents represent an interesting paradigm to develop intelligent and distributed systems, because of their autonomy, proactiveness and reactivity; in addition, their sociality enables the distribution of the application logic in different agents that can interact together and with the host environment. In such scenario interactions must be carefully designed and managed at run-time. The concept of role has been adopted in different (agent) approaches to flexibly manage interactions; roles represent stereotypes of behavior, which are useful not only in the modeling of systems and applications, but also in their design and implementation. Overhearing is a technique that enables the observation of agents' behavior by "hearing" the exchanged messages. In general, overhearing can be useful to "label" observed agents and manage agent organizations. In this paper we explore the adoption of overhearing in conjunction with agent roles in order to provide more features to agents themselves. In particular the approach presented here can support and help the agent deciding the role to assume and then how to use it, after having observed the agent behavior.**

***Index Terms*— agent interactions, roles, overhearing**

1. INTRODUCTION

Distributed intelligent systems can rely on software agents, which are autonomous software entities with interesting features. First, they are autonomous and adaptable problem-solving entities, able to execute in open and dynamic environments and to carry out their task(s) without requiring a continue user involvement [17]. In fact, because of their autonomy, agents can play on behalf of their owner, resulting as a digital counterpart of the latter in a digital world. Moreover, due to their adaptability, agents can face dynamic and exceptional situations, representing a robust and reliable approach to build complex software systems [24]. Another important characteristic of agents is mobility, which enables agents to move themselves across the hosts of a network; these agents are called *mobile agents*. Mobile agents are suitable to play on behalf of their user since, as the user do in the real world, they can move in the digital world searching for and reaching required data, visit other sites, meet other agents and exchange information with them.

Applications based on agents often involve more than one agent, even mobile, in order to divide complex tasks into smaller ones; such kind of applications are called MAS – Multi Agent Systems. In MAS applications, interaction between agents must be carefully modeled, since they are fundamental for the whole system, in terms of both "distribution" and "intelligence". In fact, the distribution of the applications can occur by spreading agents, which can carry out the applications' task(s) by interacting each other; these interactions can exhibit a given degree of intelligence also with the support of the underlying environment.

So far, several approaches have been proposed to manage and face agent interactions and organizations, including Tuple-Spaces [11], Group Computation [15], Activity Theory [20], Roles [12] and Overhearing [3, 4, 16]. Our work takes into consideration the last two (Roles and Overhearing), and this paper proposes a few considerations to merge the two approaches in order to enrich the Role-based one. The Role theory proposes to manage agent collaborations and interactions through the use of stereotypes of common behaviors, resulting in a powerful and easy to understand way to model interactions. The Overhearing theory proposes to manage agent collaborations through a multicast listening to all exchanged messages. Both the former theories propose to manage interactions between agents, but providing different approaches that can be combined to enhance and better customize the interaction management.

The key idea of this paper is that it is possible to overhear an agent (i.e., to observe its behavior), in order to suggest that it assumes a role and uses it to carry out its tasks. To do this, it is not enough to apply the overhearing theory as it is, to a role-based agent context: overhearing must be adapted in order to achieve a smart monitoring system, able to observe the behavior of either single agents or group of agents.

The paper is organized as follows: Section 2 glances at background of Roles and Overhearing theories and motivates a combined approach; Section 3 presents a few drawbacks of pure overhearing, while Section 4 explains how our approach works; Section 5 provides a few application examples; Section 6 proposes some implementation directions, and finally section 6 concludes the paper.

2. BACKGROUND & MOTIVATIONS

This section introduces both the theory of Roles and Overhearing, giving a brief background of both, and showing how enhancing roles with overhearing can result in a support system for agent decisions and proactiveness.

2.1 Roles

The Role theory [2] has been applied to several computer science fields, and in fact there are several

Manuscript received October 16, 2006; revised January 19, 2007 This work was supported by the EU under the CASCADAS project and by the Italian MUR under the "Agent oriented methodologies: engineering of interactions and relationship with the infrastructures" project. This paper is extended from "Collaboration-Driven Role Suggestion for Agents" published at *IEEE Workshop on Distributed Intelligent Systems Collective Intelligence and its Applications*, Prague, CZ, June 2006.

G. Cabri, L. Ferrari, L. Leonardi and R. Quitadamo are with Dept. of Information Engineering, University of Modena and Reggio Emilia, via Vignolese, 905, 41100 Modena, Italy (e-mail: giacomo.cabri@unimore.it, luca.ferrari@unimore.it, letizia.leonardi@unimore.it, and raffaele.quitadamo@unimore.it).

definitions of the concept of *role*, depending on the considered scenario. In particular, it has been noted that, during the Object-Oriented design phase, there are classes that are not really classes, but roles [22]. Roles represent a cross-cut view of the object space, and thus can be adopted to model dynamic and open environments. Roles are typically related to associations among software entities, as emphasized in the UML language, where roles are applied to associations between objects [23].

The interesting feature of roles is that they can be used as a paradigm to smartly model the view of a complex system [13]. This is of primary importance when roles are applied to the agent world; in fact, recalling that agents can be thought as human counterparts, roles represent a conceptual tool to model the digital world in a way similar to the real one, where digital human counterpart can perform their task(s) in ways similar to those adopted by humans in the real world organizations and collaboration [25].

More in detail, roles can be applied to agents in order to both enhance their capabilities, granting a better adaptability, and to model interactions and coordination in MAS systems [12]. For example, consider an agent in charge of writing a few records into a database. By placing the code in charge of interacting with the database into a specific role, let it be the *writer*, the agent playing such role is free to discard any detail regarding the database system and its interface. In this way, just by assuming and playing the writer role, the agent (see Figure 1) can perform the record insertion without knowing any detail about the underlying database system. The use of the writer role enhances the agent capabilities, since, without that role, it is unable to insert the records in the database, and, at the same time, the writer role grants to the agent a stronger adaptability. In fact, if the database changes, it suffices to change only the role, without applying any change to the agents that will play it.

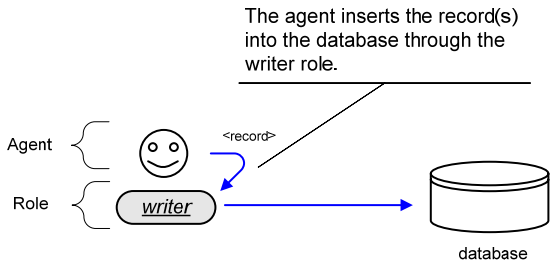

Figure 1. An agent that exploits a role to achieve its task.

As shown in Figure 2, roles can be also used to manage interactions between agents. In fact, roles can be developed with regard to each other, thus the *bidder* and *seller* roles of Figure 2 can implement a common protocol in order to let the related agents playing them make an offer and sell a good.

It is important to note that roles are tied to the local execution environment, thus they represent context-dependent views of entities running in that environment [1], granting adaptability. With regard to the above example, this means that the bidder and seller roles belong to the local interaction context the agents are running in, and then if the agents move to another interaction context (e.g., another host), local roles can have a different implementation. It is for this reason that we claim that roles grant portability and generality: since they are tied to each interaction context, they hide context details to agents, which are free to discard those "low-level" details.

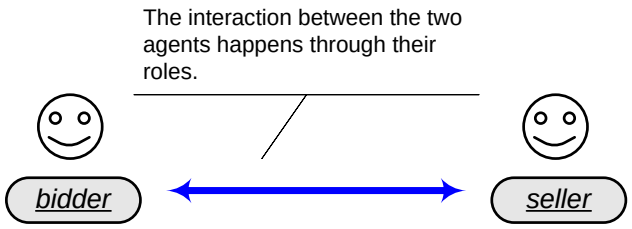

Figure 2. An interaction through the use of roles.

In order to play a role, an agent must *assume* it. In other words, an agent must choose a specific role, that means that the role assumption is considered an *active* process; but there exist some approaches where the role assumption is a *passive* approach, such as *overhearing*.

2.2 Overhearing

Overhearing [16] theory can handle group of collaborating agents that auto-coordinate themselves in order to reach a common aim. The key idea is that it is possible to classify, assigning a specific role, each agent simply listening to it, that is observing its message exchange with other agents [4]. Capturing and understanding the messages that agents are exchanging, it is possible to infer which role the one is playing with regard to the other; thus it is possible, for example, to understand which agent is playing the role of seller and which is playing the role of bidder in the example of Figure 2.

Typically, overhearing is used to manage groups of agents that agree on a common aim. These groups, often called *implicit organizations* [5], are made by agents that, thanks to the overhearing process, understand that they have a common task or aim; therefore, they aggregate to perform such task and to synchronize. In each group, a leader agent (called *oracle*), emerges and drives the group to achieve the final aim.

The main difference between the role concepts used in both the above theories is the following: while, according to the role theory, a role is an entity that must be actively assumed, in the overhearing theory it is a passive classifier, used only to recognize the agent behavior, instead of granting a specific behavior, as in the role theory. In other words, while, in the overhearing approach, roles are mainly used to passively get a bird's eye view of the agent scenario, in the role theory they are actively used by agents to enhance their capabilities.

Starting from the above considerations, it could be seen that the role theory is, in general, more appropriate for agents, since they are active entities. Moreover, it could seem that the two theories cannot coexist, since overhearing uses the role concept as a passive classifier, while the role theory uses it as a set of enrichments for the agent that has assumed it.

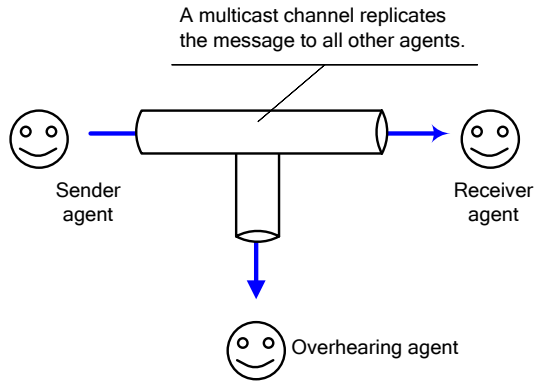

Figure 3. Use of multicast channel.

Please note that the overhearing process will not run for all the agents' life, but only until the implicit organization has been created. After that, agents that joined the organization can stop overhear each other until a new interaction context or the need for another organization arises.

3. OVERHEARING VS. ROLES

In our opinion, the overhearing theory cannot efficiently support a dynamic and open environment, such as that where agents can be executing in. It must be pointed out that overhearing requires that all messages incoming and outgoing from a set of agents are caught and analyzed. In order to achieve this, agents can use *multicast channels* [3], which are ad-hoc communication links that replicate a message to multiple listeners, as shown in Figure 3.

The use of such channels presents a few drawbacks in the overhearing theory. First of all, the fact that an agent can listen to a communication between two other agents can lead to security problems, since the privacy of the latter two agents is broken. This is emphasized by cases where the listening agent will not, or cannot, participate to the interaction that occurs between the two other agents. For example, consider the case of a bidder agent and a seller agent in a sealed auction; in this case, a third bidder agent should not be able to listen to the bid made by the first one, or the auction semantic will not work (bids should be secret, in order to rightly evaluate a winner). Another example could be the case of a listening agent, playing as a database writer, that is not interested in the auction context; in this case the information listened by this agent is useless, thus it is wasting its resources. Moreover, messages can be spoofed, arising problems in the interpretation of messages themselves and of the behavior of their emitters.

One more disadvantage of the overhearing theory is that it is supposed to work through a finite state machine, adopted to recognize the communication happening among agents starting from the exchanged messages. Of course, since overhearing is based on the listening of exchanged messages, there can be situations where a listening agent cannot understand the running transaction due to the lost of a few messages. Moreover, it is possible that an agent exchanges a very few messages with regard to its internal changes; in such situation, it becomes very difficult to understand what is happening through overhearing. In addition, the interpretation of a message could be too much subjective, that means that different agents could not agree on the meaning of a specific multicasted message. This can lead to situations where the aggregation of agents with the same common aim (implicit organizations) never happens, just because agents cannot agree on the meaning of the exchanged messages. Finally, agents developed to exchange a lot of messages can lead to an overloaded system, where the most of the resources of agents are busy interpreting a multicasted message instead of performing the real aim of the agents. In other words, the overhearing theory will not scale very well in environment running a lots of agents, while the adoption of roles will not produce the same problem since roles are expressively designed to be applied to only one interaction context.

It must be pointed out that overhearing is almost a passive approach, since an agent must wait for incoming multicast messages in order to analyze and understand what is happening in the surrounding environment. This is in contrast with the role theory, which is primarily an active approach, where an agent is supposed to autonomously and actively perform one or more role action(s). Moreover, the role theory allows a modular development and a high concern separation [8]: roles to be played in the same interaction context are supposed to be developed together (or at least the one with regard to the other) and in a separated way from other role/interaction contexts and even from the agents that will use them. Instead, the overhearing approach cannot result very modular, since it is not well known a-priori which messages agents are going to listen and how the interactions are going to appear.

Finally, the overhearing theory tends to classify and couple agents depending on their aim, while the role theory tends to couple agents depending on the interaction context disregarding their aim(s) (that could even be competitive, especially in an open and wide environment).

Starting from the above considerations, we emphasize how the overhearing theory is not appropriate for very dynamic and open environments, while the role theory is. However, we believe that overhearing can be exploited to enhance role approaches helping agents deciding which role to assume simply listening (observing) the ongoing interactions. Overhearing can be achieved without requiring any change to the agents, that means that legacy

code can be kept while overhearing comes with an initial zero cost. Furthermore, the knowledge of how agents are executing (i.e., which behavior they are exhibiting) in a context makes possible to implement strategic decisions. Finally, observing which agents have the same behavior can be useful to implement a redundant system.

It is for the above reasons that we decided to propose a role approach enhanced by overhearing, and the following section details how it works.

4. COMBINING ROLES AND OVERHEARING

Role approaches relies on a so-called *role system*, which is an implementation of the approach itself [8]. The role system drives the assumption and release of a role, presenting also the list of available roles to a requesting agent. On the one hand, the role system is in charge of defining the mechanisms behind the role assumption and release, that is how physically the role becomes owned by an agent and how the latter dismisses the role. On the other hand, the role system drives also the choices of the roles, for example denying the assumption of incompatible roles by the same agent (e.g., an agent that wants both the bidder and seller role for the same auction). The adaptability provided by the role system is fundamental in order to cope with open environments [7, 9].

Our approach is oriented to open environments, where agents are supposed to exhibit a high degree of interaction, needed to carry out their tasks. So we assume not only that agents are enabled to interact by exchanging messages, but also that interactions are only limited by security reasons.

It is possible to apply overhearing at the level of agents, which means that agents perform overhearing by themselves, or at the role system level, where the *system* does overhear the agents. In order to keep agents simple and do not overload them, our approach exploits overhearing at the role system level. In other words, the overhearing is performed by the role system itself. This choice has a few advantages: first, the role system can apply the same semantic rules to all caught messages, removing any "subjectiveness" that agents could introduce. Moreover, this choice enables the environment to become more complete and to support simple agents, like in eco-environments applications [18].

Furthermore, the role system has a "panoramic view" of the whole system during its life, which means it can handle all messages without losing any piece of useful information. Thanks to that, the role system can help agents suggesting other roles they need to aggregate in groups, or simply providing a way to contact counterpart agents. Finally, this solution does not require broadcasting any message, since they are caught only by the role system, which is a part of the surrounding environment. This means that (i) the messages are caught by a trusted entity and (ii) there are not secrecy or privacy problems.

The role system can analyze information coming from the observed messages and send then other messages (or events) to interested agents. This will help them achieving their task(s). In other words, the role system plays as a unique coach of several teams of agents, coordinating each single team and even coordinating teams between them.

It is worth noting that, while in simple overhearing it is important to decide how and to which agents to assign multicast channels, that is which agents must be listened to, leaving the role system to overhear does not require this decision. At least, at the beginning, the role system can decide to listen to all agents, and depending on its intelligence, it can decide, for example, to stop overhearing some agents, decreasing the number of caught messages. Anyway, the role system will always keep the dynamic situation of all agents.

We remark also that our approach is not intended to check if the exchanged messages comply with environment laws, such as in [21] and [19] where the aim is to deny interactions forbidden by local laws, even if local rules of interactions can still be enforced since agents must interact according to played roles [6]. In addition, our approach satisfies the requirements of agent autonomy [14], since it does not imposes the assumption of roles, but only suggests it to agents.

Please note that the adoption of overhearing in a role system does not guarantee the correct suggestion of the right role to each agent, but simply provides a way to better manage choices, resulting also in a good workbench for agent behaviors and static analysis.

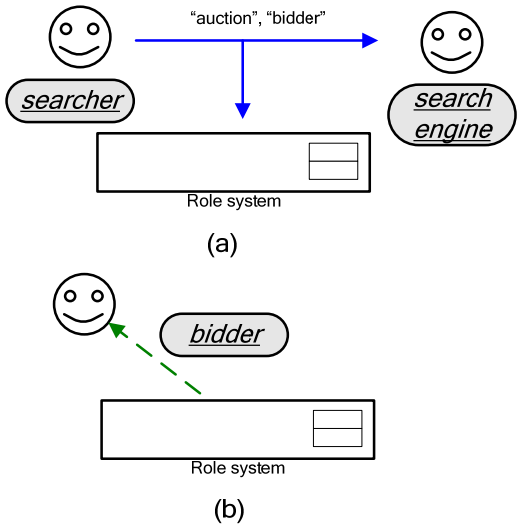

Figure 4. An agent searches for a service by providing keywords (a) and the role system suggests assuming an appropriate role (b).

5. RUNNING EXAMPLES

This section introduces some examples in order to show how an overhearing role system can be used to suggest and support the agent execution and the role choice.

The first example is quite trivial. Suppose that an agent exploits a search engine to find interesting services or entities in general (Web sites, other agents, etc.). In this case, an appropriate *searcher* role is assumed by the agent.

The role system can overhear the interaction between the agent and the search engine, and analyze the keywords supplied by the agent (see Figure 4 (a)). If these keywords match some keywords of available roles, then the latter ones are candidate to be suggested to the agent, which not only find out services/entities, but also is provided with appropriate roles to interact with them (see Figure 4 (b)).

In a second set of examples, we consider the already briefly explained example of an auction, where there are several agents playing as bidders, one (per time) playing as seller and one playing as auctioneer. First of all, embedding each specific behaviour into a single role (thus, there are the *bidder*, *seller* and *auctioneer* roles), each agent must assume one of the above role to participate to the auction. The first case, tailored over the above explained auction simulation, regards an agent that needs to sell a good in order to make money. The first thing this agent can autonomously do is selecting and then assuming the seller role. But what happens if the auction related to that kind of good has not been started yet? This means that, for example, there is no agent playing the auctioneer role for that kind of good yet. In this situation, the seller agent cannot make money, and its execution is blocked. Now imagine that an agent assumes the auctioneer role; since the role system, having overheard, knows that there is a bidder agent waiting, and an auctioneer not assigned to a specific auction yet, it can suggest the auctioneer to start an auction of that specific kind of good, and then notifying the seller agent that there is an auctioneer. A similar case, that implies a stronger use of overhearing, happens when the role system understands that an auctioneer is closing its auction, thus it suggests this auctioneer to start another auction for the kind of good the seller agent is waiting for. A different case happens if the role system intercepts a message of a bidder requesting for a specific good, which has been acquired before from another bidder agent. In this case, the role system can suggest to the second bidder to change its role into the seller one, and to sell the good to the other bidder. It is important to note that, in the above examples, the role system can do more than suggesting the next role to assume to agents; the role system can know which agents must be connected, because the role system detects that they are exchanging messages. Therefore, the role system can easily and quickly provide the address of counterpart agents, in order to speed up and simplify interactions.

Another example involves an agent (A1) that wants to acquire a good (G1), and thus has assumed the *bidder* role and has made its offer. Imagine that the agent has not enough money to get the good, thus its offer is refused. Imagine also that the agent A1 owns a good (G2) that is subject of interest for a second agent (A2), which has just joined the environment assuming the *bidder* role. Since the role system knows, from the occurred interactions, that (i) the bidder A1 has not enough money and that (ii) the bidder A2 is interested in buying the good G2 of the other agents, it can suggest the bidder A1 to temporary assume the *seller* role in order to sell its good to agent A2, making money and increasing its possibility to acquire the good G1. Of course the role system does not know (or has a very few chances to know) that A1 owns G2, which is the good A2 is interested in. However the system can first suggest to A1 to assume the seller role and can observe interactions that involve A1. If there is a successfully transaction between A1 and A2 it means that A1 owned the G2 good and the made prediction was right. This could lead to the situation where another bidder (A3) joins the environment, requiring the same good G2 and the environment can quickly suggest the appropriate role(s) to interact with the agents that are likely to own the interested good.

Role suggestion can be useful also in tampered environments: imagine that, disregarding all designer and developer efforts, an environment presents roles that are not rightly “described”. What could happen if a role *database_writer* undergoes a wrong refactoring and becomes a role that writes on plain text files? An agent will try to use the above role expecting another agent able to query the database and to find the inserted data, while the data is stored in another media. In such situation the role system can observe the behaviour of agents that exploit the above role, noting that the role actions are not coherent with their “description” (or, better, with their descriptor as reported in [8]) and could (i) notify the administrator about a possible problem and, at the same time (ii) suggest agents that require such role to choose another one or to be aware of the possible problem.

As a last example consider the following: some agents are writing, independently, a few records in the database. In this case, as already written in section 2, it is better to use a *writer* role as abstraction over the database backend. Imagine that there are different agents playing the writer role at the same time, and that the database is going overloaded. Here the role system can suggest new agents that are requesting the writer role, to change the database (e.g., move to another host), or to use another role, connected to another database. Supposing then that it is possible to use different writer roles, with different privileges and priorities with regard to the database access, thus during an overloaded situation the role system can recognize and suggest the right privilege level to each agent (i.e., the more appropriate writer role to assume). This example in particular emphasizes a situation that is difficult to achieve with pure overhearing, since here each writer agent is independent from each other, that means they are not exchanging any message and do not represent a group or an organization.

As readers can see from the above examples, the overhearing is exploited within a role system not to primarily find out the main aim of a group of agents (composing thus an implicit organization), rather to suggest each agent the right interaction context (i.e., the set of roles to use) depending on an external point of view of its behaviour.

6. SYSTEM IMPLEMENTATION

In this section we sketch some hints for a possible implementation of the proposed approach. We will address two main issues: how to overhear the exchanged messages and how to suggest roles to agents.

We refer to the RoleX infrastructure [9, 10] to sketch some proposals for the implementation that combines roles and overhearing. RoleX is an infrastructure devoted to manage roles for agents. Rolex is a role-based middleware and therefore does not substitute the agent platform, which is still in charge of providing the services for the agents' life cycle; instead RoleX provides additional role-related services: it allows the assumption and release of roles and enables the interaction between agents playing given roles.

In RoleX, all interactions are managed by the underlying role systems, in an *action-event* fashion: an agent selects an action among the available from the played role, and the role system translates this action into one or more events that are delivered to the counterpart entity (often another agent). For instance, in an auction, the bidder agent makes a bid via the *makeBid* action, which implies the delivering of an event to the auctioneer agent, which checks and registers the bid. The fact that the underlying role system manages the interaction in this way makes it possible to observe the interactions between agents playing roles. This well suits the overhearing aim of our approach.

With regard to the suggestion of roles, again the event mechanism can be exploited to this purpose. RoleX itself uses events to communicate to running agents special conditions or evironment changes, thus it is possible to enable RoleX suggesting a role by means of events. Such events will be sent by the RoleX environment itself to agents, which will then evaluate and decide consequently how to proceed (in the case simply discarding the events). It is important to note that RoleX fully manages the role assumption/release process for each agent running on top of the middleware. This means that, in the case an agent follows the suggestion of RoleX assuming a specific role, the middleware itself becomes aware of it, being then able to refine suggestions to other agents.

7. CONCLUSIONS AND FUTURE WORK

To consider agents a suitable paradigm for distributed intelligent system, their interactions must be carefully faced and designed; in this paper we propose to combine the Role and Overhearing theories to support agent interaction in an adaptable way. On the one hand, roles are useful to model interactions and to manage cooperation in agent-based applications. On the other hand, overhearing is a powerful tool to dynamically manage agent aggregations and to try to recognize what agents are doing, but it suffers from a few drawbacks that make it not appropriate to be applied in open and dynamic MAS systems.

Combining the activeness of the Role theory and the passiveness of the Overhearing approach, it is possible to get advantages from both theories: the role theory lets agents be as much active as possible, while overhearing can help suggesting agents the roles to assume, growing the adaptability to the scenario.

Our overhearing role system can help agents adapting in a faster way to the environment where they live: since the system can suggest roles to agents, it can help agents assuming the best role to assume and use to interact with the other agents or entities in the environment.

With regard to future work, we sketch some directions.

First, the suggestion of the roles is the most delicate issue to be faced. In fact, on the one hand a suitable way to suggest roles must be designed in order to be effective and useful for the agents; the one sketched in section 6 could be a starting point, but different role-based systems may have different requirements. On the other hand, it should be as less intrusive as possible, without forcing agents to assume roles preserving agents' autonomy.

Then, even if exploiting role systems for overhearing reduces privacy issues, appropriate mechanisms should be enforced to let agents trust the underlying role system and allow it to hear the exchanged messages. In connection with this point, role systems should consider also situations where privacy motivations are stronger than overhearing advantages, and private communications between agents have to be enabled.

As already mentioned, for the implementation of the proposed approach, we are going to exploit RoleX [9], a role-based middleware that can enable the overhearing of agents' interactions. Thanks to the capabilities of RoleX, that provides an event communication system even on those agent platforms that do not use events, allows us to exploit events as a portable way for role suggestion.

ACKNOWLEDGEMENT

The authors would like to thank the anonymous reviewers and editor for provided comments, which helped in improving this paper significantly.

REFERENCES

[1] D. Baumer, D. Ritchie, W. Siberski, M. Wulf, "The Role Object Pattern", *Proceedings of the 4th Pattern Languages of Programming conference (PLoP)*, Illinois, USA, Sep. 1997

[2] B. J. Biddle, and E. J. Thomas, *Role Theory: Concepts and Research*, New York, R. E. Krieger Publishing Co., 1979.

[3] P. Busetta, M. Merzi, S. Rossi, F. Legras, "Intra-Role Coordination Using Group Communication: A Preliminary Report", *Proceedings of the International Workshop on Agent Communication Languages and Conversation Policies (ACL2003)*, Melbourne, Australia, July 2003. Springer, LNAI 2922, pages 231-253

[4] P. Busetta, S. Rossi, "Towards Monitoring of Group Interactions and Social Roles via Overhearing", *Proceedings of the Eighth International Workshop on Cooperative Information Agents (CIA 2004)*, Erfurt, Germany, September 2004. Springer-Verlag, LNAI 3191, pages 47-61

[5] T. Kuflik, P. Busetta, L. Penserini, P. Bresciani, M. Zancanaro, "Personalized Information Delivery in Dynamic Museum Environment by Implicit Organizations of Agents", *Proceedings of the Workshop on Environments for Personalized Information Access*, Gallipoli, Italy, May 2004.

[6] G. Cabri, "Agent Composition via Role-based Infrastructures", *Scalable Computing: Practice and Experience*, Vol. 7, No. 1, pp. 37-47, ISSN: 1895-1767, March 2006.
[7] G.Cabri, L.Ferrari, L.Leonardi, "BRAIN: a Framework for Flexible Role-based Interactions in Multiagent Systems", *Proceedings of the First European Workshop on Multi-Agent System (EUMAS)*, December 2003, Oxford, UK
[8] G.Cabri, L.Ferrari, L.Leonardi, "The Role Agent Pattern: a Developers Guideline", *Proceedings of the 2003 IEEE International Conference on System, Man and Cybernetics*, October 2003, Washington D.C., U.S.A.
[9] G. Cabri, L. Ferrari, L. Leonardi, "The RoleX Environment for Multi-Agent Cooperation", *Proceedings of the 8th International Workshop on Cooperative Information Agents (CIA)*, Erfurt, Germany, September 2004, Lecture Notes in Artificial Intelligence N. 3191, Springer-Verlag.
[10] G. Cabri, L. Ferrari, L. Leonardi, "Injecting Roles in Java Agents Through Run-Time Bytecode Manipulation", *IBM Systems Journal*, February 2005, Vol. 44 N.1, pp.185-208
[11] G. Cabri, L. Leonardi, F. Zambonelli, "MARS: a Programmable Coordination Architecture for Mobile Agents", *IEEE Internet Computing*, Vol. 4, No. 4, pp. 26-35, July-August 2000.
[12] G. Cabri, L. Leonardi, F. Zambonelli, "Implementing Role-based Interactions for Internet Agents", *Proceedings of the 2003 International Symposium on Applications and the Internet (SAINT 2003)*, Florida, USA, January 2003
[13] M. Fowler, "Dealing with Roles", http://martinfowler.com/apsupp/roles.pdf, 1997.
[14] A. Gouaich, "Requirements for achieving software agents autonomy and defining their responsibility", *Proceedings of the workshops on Agents and computational autonomy: potential, risks, and solutions at AAMAS 2003*, Melbourne, Australia, July 2003, Lecture Notes in Computer Science, Springer, Vol. 2969.
[15] B. Hirsh, M. Fisher, C. Ghidini, "Programming Group Computations", *Proceedings of the First European Workshop on Multi-Agent System (EUMAS)*, 18-19 December 2003, Oxford, UK
[16] G. A. Kaminka, D. V. Pynadath, M. Tambe, "Monitoring Teams by Overhearing: A Multi-Agent Plan-Recognition Approach", *J. Artif. Intell. Res. (JAIR)* 17: pp. 83-135 (2002)
[17] M. Luck, P. McBurney, C. Preist, "Agent Technology: Enabling Next Generation Computing – A Roadmap for Agent Based Computing", AgentLink, http://www.agentlink.org/roadmap
[18] P. Marrow, M. Koubarakis, R.H. van Lengen, F. Valverde-Albacete, E. Bonsma, J. Cid-Suerio, A.R. Figueiras-Vidal, A. Gallardo-Antolín, C. Hoile, T. Koutris, H. Molina-Bulla, A. Navia-Vázquez, P. Raftopoulou, N. Skarmeas, C. Tryfonopoulos, F. Wang and C. Xiruhaki, "Agents in Decentralised Information Ecosystems: the DIET Approach", *Proceedings of the Artificial Intelligence and Simulation Behaviour Convention 2001 (AISB'01), Symposium on Information Agents for Electronic Commerce*, pp. 109-117, York, UK, March 2001.
[19] N.H. Minsky, V. Ungureanu, "Law-governed interaction: a coordination and control mechanism for heterogeneous distributed systems", *ACM Transactions on Software Engineering Methodology*, 9 (3) 273-305, 2000.
[20] A. Omicini, A. Ricci, S. Ossowski, "Rethinking MAS Infrastructure based on Activity Theory", *Proceedings of the First European Workshop on Multi-Agent System*, December 2003, Oxford, UK
[21] R.Paes, G. R Carvalho., C.J.P. Lucena., P. S. C.Alencar, H.O. Almeida, and V. T. Silva, "Specifying Laws in Open Multi-Agent Systems". *In: Agents, Norms and Institutions for Regulated Multi-agent Systems (ANIREM)*, AAMAS2005, 2005.
[22] F. Steimann, "Role = Interface: a merger of concepts", *Journal of Object-Oriented Programming*, Vol. 14, No. 4, pp. 23-32, 2001.
[23] *Unified Modeling Language Specification*, Ver. 1.4. OMG.
[24] M. Wooldridge, N. R. Jennings, D. Kinny, "The Gaia Methodology for Agent-Oriented Analysis and Design", *Journal of Autonomous Agents and Multi-Agent Systems*, Vol. 3, No. 3, pp. 285-312, 2000.
[25] H. Zhu and M.C. Zhou, "Role-Based Collaborations and their Kernel Mechanisms", *IEEE Tranactions. on Systems, Man and Cybernetics*, Part C, vol. 36, no. 4, July 2006, pp. 578-589.

Giacomo Cabri is an associate professor in Computer Science at the University of Modena and Reggio Emilia, since 2006. He received the Laurea degree in Electronic Engineering from the University of Bologna in 1995, and the PhD in Computer Science from the University of Modena and Reggio Emilia in 2000. His research interests include methodologies, tools and environments for agents and mobile computing, wide-scale network applications, and object-oriented programming.

Luca Ferrari got his PhD in Computer Science at the University of Modena and Reggio Emilia in 2006. He received the Laurea degree in Computer Science Engineering from the University of Modena and Reggio Emilia in 2002. His research activity covers the study of models, methodologies and environments for mobile agent systems, object-oriented programming and Java-based technologies.

Letizia Leonardi is a full professor in Computer Science at the University of Modena and Reggio Emilia, since 2001, where she teaches basic and advanced computer science courses. She received the Laurea degree in Electronic Engineering in 1982 and the PhD in Computer Science in 1989, both from the University of Bologna. Her research interests include design and implementation of coordination infrastructures for mobile agent systems, object-oriented programming environments, and parallelism and distribution issues, especially as they apply to object systems.

Raffaele Quitadamo is a PhD student in Computer Science at the University of Modena and Reggio Emilia. He received the Laurea degree in Computer Science Engineering from the University of Modena and Reggio Emilia in 2004. His research interests concern mobile agent and thread systems, service-oriented infrastructures and autonomic computing.