Rethinking Agent Roles: Extending the Role Definition in the BRAIN Framework*

Giacomo Cabri, Luca Ferrari, Letizia Leonardi
Dipartimento di Ingegneria dell’Informazione
Università di Modena e Reggio Emilia
Via Vignolese, 905 - 41100 Modena - Italy
{cabri.giacomo, ferrari.luca, leonardi.letizia}@unimo.it

Abstract – Agents represent a technology that grants developers a new way to develop complex applications. Thanks to their autonomy, reactivity and mobility agents can be exploited in today’s applications, even playing on the behalf of users. Agents are not really useful if isolated, instead their powerful is increased as much as they can cooperate and coordinate with other agents or environments. To deal with the need of coordination, developers can use the role-based approach, where coordination issues are embedded in roles exploited by agents. Nevertheless, up to now, agent roles have been entities that agents can exploit, without following and providing a real agent evolution, as happens in the real life for human being. In this paper, we propose work in progress within the BRAIN role-based framework to overtake the above limitation, providing a new definition of role.

Keywords: roles, agent interactions, evolution.

1 Introduction

Agents are autonomous, problem-solving, adaptable computational entities that can perform their task(s) in open and dynamic environments [16]. The main characteristic of agents is autonomy, which enables them to perform actions without requiring a continue user involvement. Furthermore, autonomy enables agents to act on behalf of their user in a digital world, thus they can be thought as human counterpart in that digital environment.

Another important characteristic of agents is mobility, which enables agents to move themselves among the hosts of a network. Agents able to move are commonly called mobile agents. Mobile agents can better play on behalf of their user since, as the user can 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 exchanging information with them.

Today’s agent-based applications often involve multiple agents in order to divide complex tasks into smaller ones; such kinds of applications are called MAS – Multi Agent Systems. In MAS applications coordination among components (i.e., agents) plays a fundamental role. There are several approaches related to agent coordination, including Tuple-Spaces [7], Group Computation [14], Activity Theory [17] and Roles [8].

In this paper, we propose some considerations about the role-based approach. Since we think that agents should effectively be able to play as digital human counterpart, we believe that the paradigm should be inspired as much as possible to the one adopted in the real life. For this reason a role should be not only a “classifier”, but should be able to increase the agent capabilities and knowledge, allowing it to evolve itself in the current environment. During their life agents can evolve, that means they become able to do more tasks than those they are developed for. This evolution is similar to what happens in the real life, where humans learn how to do new things and expand their knowledge. Similarly to the real life, where the evolution can lead to role changes, also in the agent world this should happen. Moreover, during their life agents can change their role not only accordingly to what they are going to do, but also to what they are able to do. Starting from the current BRAIN framework [3] we propose a way to adapt the role paradigm to the above considerations in order to reflect, through roles, the agent evolution, making the adoption of such paradigm more flexible.

The paper is organized as follows: section 2 glances at background and motivations for the new definition, section 3 introduces the BRAIN framework; section 4 details changes to the definition of the concept of role. In order to better explain presented concepts, section 5 reports an application example, while section 6 reports related work. Finally the conclusions end the paper.

2 Background & Motivations

There are several definitions of the concept of role, depending on the field where they are used; several of them start from the Object Oriented Programming. In fact, it has been noted that, during the OO design phase, there are classes that are not really classes, but roles [19]. Unlike classes, roles are more dynamic, changing at runtime and meeting better the application requirements.

Roles are typically related to associations among software entities; we can find a definition of the concept of role even in the UML language, where roles are applied to associations between objects [21].

Roles can be used as a paradigm to model in a better way the real world and the vision that developers have of it [13]. For this reason roles can be used to model and build a digital world 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.

Roles can be applied to agents both to expand agent capabilities, granting adaptability, and to model...
interactions allowing an easier coordination among MAS applications [5, 6]. Since roles can be though as context-dependent views of a particular object or entity [2], they can be used to build interaction models specifying each involved-part characteristic. Nevertheless, interactions can hardly be managed in a global space, and for this reason it is important to define local interaction contexts to which roles are applied.

A local interaction context is an environment where agents can interact; the context defines rules and possible associations among agents; roles belong at least to one context. From the agent point of view, the context is viewed in a subjective way, that means that agents feel the context as a suitable representation of the environment where they live and act. From the designer/developer point of view, instead, the context is viewed in an objective way [17], which means as a set of mechanisms to coordinate agents modeling their interactions. Roles can cover the difference between the two views, helping developers understanding how agents become aware of the context, leading thus to a better development of complex agent applications.

Unlike OOP, where roles are mainly used as “classifiers”, in the agent scenario they embed context-dependent details, such as interaction protocols or facilities to perform a particular task, so that an agent playing a specific role can exploit it to achieve its aim. For instance, as shown in Figure 1, the agent can insert a tuple into a tuple space [7] exploiting a particular role, called tuple_writer.

Figure 1. An example of agent exploiting its role to perform an operation

Another example is shown in Figure 2, where an agent playing the role of manager is coordinating the actions of one or more agents playing the role of employee. In both the above situations roles are used to perform actions without knowing details about them. This means that roles embed context-dependent details, for example either the right instruction to insert a tuple in the tuple space of Figure 1, or the communication protocol used in Figure 2. This leads to a stronger adaptability of agents, which can disregard the details about the interaction context, since they are embedded in the roles. This is really useful especially with mobile agents since, while moving, they can need to act (and interact) in possible different context implementations, depending on the hosts in which they are executing.

Agents can exploit roles in a dynamic way, in order to act as coordinating entities and not as coordinated entities. This means that, thanks to their autonomy and the capabilities granted by roles, agents can coordinate themselves in a proactive way, without requiring a fixed a-priori coordination schema.

Roles are available to agents through a role system, which is a system that manages roles and their use [6]. Agent can use roles after a process called role assumption, which ties the role to the agent and grants all role characteristics to the agent. The agent can dismiss the role when it is no more needed through a role release. How the assumption and release processes work depends on the role system implementation and will not be took into account here (see for example [5, 6, 9, 12]); what is important to note however is that roles are not only used as simple “classifiers” but, since agents are free to chose them accordingly to their aims, roles are a kind of enhancement for agent themselves.

3 The BRAIN Framework

BRAIN (Behavioral Role Agent INteractions) is a role-based framework that helps developers to manage interactions and coordination in MAS application. As shown in Figure 3, the framework is composed of three layers. The higher level is a Role-based model, which defines the interaction context and roles involved in the context. The middle layer is a XML-based layer, which defines a XML notation (called XRole) to describe roles using XML documents. The use of XML enables developers to use the same role description with different systems, even built using different technologies, thanks to the XML portability. Furthermore XML allows developers to obtain several views of the same information (the role description) through XML transformation (e.g., using XSL).

Finally, the last layer is composed of (possible) several Interaction-Infrastructures, which represent the role system itself and which combine the description of the XML notation and of the role based model into
available role based mechanisms. Since each interaction infrastructure represents the role system, it defines how the assumption and release mechanisms work.

BRAIN defines a role as a set of capabilities, knowledge and expected behavior, which the agent can assume, use and release at run-time in order to achieve its aims [5]. In other words, on the one hand a role extends the agent capabilities granting new available actions and, on the other hand, the agent is required to manage the assumed role, thus showing the expected behavior. For example, with regard to Figure 2, the manager role adds to its owner the capabilities to communicate tasks to the employee role (capabilities), while the employee role manages interactions with the manager (expected behavior).

Figure 4. Agent capabilities changes immediately after a role assumption

Even if in the BRAIN framework roles can be used (i.e., assumed, exploited and released) in a dynamic way, they present a discontinuity between the time just before the assumption and the time just after it. In fact, before the role assumption, the agent A does not have role capabilities and behavior, while just after it has them. For instance, as shown in Figure 4, the agent A does not know how to coordinate the agent B (employee) without playing the manager role (e.g., because it does not know the communication protocols), while after the role assumption the agent A knows how to coordinate the agent B thanks to the added capabilities.

Figure 5. Discontinuity in agent capabilities

Figure 5 shows the discontinuity in agent capabilities due to the role assumption. Similarly the same thing happens for all other role characteristics, such as the expected behavior and the knowledge added to the agent. This does not reflects the evolution that we can find in the real life, where humans learn step by step until they have all required capabilities to assume a new role. Since we want to use agents on behalf of a real user, we need to apply real life rules to the agent world. This leads to a new definition or the concept of role, which takes into account the evolution of the application and, more important, of the agent.

It is important to note that the above discontinuity is an issue related not only to the BRAIN framework but also to other role-based approaches. The next section will detail a new definition of the concept of role in order to overcome this limitation.

4 Towards a new concept of role

Starting from the above sentences we want to propose a new definition of the concept of role for the BRAIN framework. Likely the real life, a role should reflect the evolution of the entity to which it is applied, in this case of the agent. In other words a role should not only increase the set of capabilities of an agent but should also classify the capabilities that the agent had before the role assumption.

4.1 Role Starting-Requirements

To achieve this, the role assumption must be driven by starting-requirements related to the initial use of the role. These requirements must be owned by the agent so that it can demonstrate that it will be able to use the role and that it deserves the role. Let us explain this concept with a simple example: imagine an agent that is going to change its role from employee to manager (i.e., it evolves from an employee to a manager). The agent must demonstrate that it is able to use the manager role, that means it is able to play as a manager; otherwise the factory where it works could fail. In order to demonstrate that the agent can play as a manager, it must provide a set of capabilities (e.g., it must understand financial data, must be able to manage projects, etc.) that must match a set of starting-requirements needed by the assumption and the initial use of the role. Starting-requirements demonstrate that the agent is enough evolved to use the new role.

The agent is not in charge of own starting-requirements from the beginning of its life, since they can be obtained during the life of the agent itself, for example due to the use of a previous role (e.g., the employee role). Starting-requirements represent the skill/experience level that an agent must have to start using a specific role.

Without owning starting-requirements, an agent could assume a role, but it could misuse it, leading to wrong results and even damaging other agents running in the same environment. For example, a manager agent that does not know how to evaluate employees could fire the best ones. Thanks to starting-requirements, role assumptions are now continuous in the time space, thus Figure 5 changes as shown in Figure 6.

Figure 6. Continuity in the agent capabilities
Of course, the assumed role will still add capabilities and expected behavior to the agent that has assumed it; starting-requirements are only needed for an initial use of the role. Without specifying starting-requirements the agent will be only enabled to use the role, while demonstrating that it owns requirements it will govern the role itself. This emphasizes one more time the autonomy of agents and lead to a more human-like behavior.

From the above sentences it should be clear how starting-requirements are not a simple set of constraints related to the role assumption. We are not interested in a situation that denies the assumption of the manager role if the agent has never played the employee role before, but in a more flexible and realistic situation, which is the denying of the manager role assumption if the agent cannot demonstrate it knows how to use it. The fact that it knows how to use the manager role thanks to its previous use of the employee role is a detail strictly related to the application scenario. In fact, it is possible to imagine an agent that owns starting-requirements for the manager role from the beginning of its life. In such situation, the agent is granted of the capability to assume and use the manager role, since it can demonstrate it owns needed requirements. One more time this lead to a more human-like world, where a man is able to assume a role not only on the base of his/her previous one.

Moreover, starting-requirements enhance the role reusability, allowing to use the same role in different contexts changing only the assumption requirements. For example, imagine that we have two contexts: factory and university. Even if these contexts are different, it is possible to use the same role, for example employee, just changing its starting-requirements in each context. In this way it is possible to use the same semantic meaning of a role in different context, reusing both the role stereotype and its implementation.

Joining roles and starting-requirements allows dynamic situations, and provides snapshots of the capabilities and agent has and have had. Moreover, if roles of a given context are in a strict connection each other, knowing the current role can be useful to know previously-assumed roles, so to reconstruct the evolution of the agent.

Thanks to the continuity among agent capabilities (see Figure 6), roles can also be seen as set of functionalities used to solve specific problems. Recalling that an agent is, first of all, an autonomous problem solving entity [16], it is possible for it to split an assigned complex task into subtasks (sub-problems), exploiting a role to solve each sub-problem [18]. Once a sub-problem has been solved, that means that the agent has acquired new experience, it can proceed with the next problem assuming a new role. Thanks to this an agent will always become more powerful thanks to the acquired role and experience.

4.2 A New Role Definition

It is now possible to provide, in a more formal way, the new definition of the concept of role that we are going to apply to the BRAIN framework (Figure 7).

Definition: A role is a set of capabilities/knowledge and expected behavior which can be assumed, used and released, accordingly to a set of starting-requirements. The above requirements are needed to assume the role and must be matched by current capabilities of the agent. The capabilities/knowledge added by the role improve the agent ones (intended before the assumption), in the case allowing it to assume other roles. Finally the expected behavior represents a set of duties that the agent playing the role has to take into account and that other agents (playing other roles) can rely on during interactions.

Figure 7. A role in the new BRAIN definition

It is important to note that the starting-requirements are embedded in the role itself in order to ease a check on the couple role-agent. In fact, it is possible to check what are the minimum capabilities of an agent simply checking requirements of its role, thus to understand what other roles the agent can assume and what tasks it can currently perform. We talk about “minimum” capabilities because starting-requirements of a role can represent a subset of those owned by the agent itself, as they represent capabilities that an agent surely owns.

4.3 Formal Notation

We can use a formal notation to describe the above concepts. Let $R$ be the set of all available roles at a specific host; a role $r_i$ is then described by a tuple $r_i = <Q_i, C_i, B_i>$ where $Q_i$ are the starting-requirements associated to the role, $C_i$ is the set of capabilities granted by the role assumption and, finally, $B_i$ is the expected behavior that the agent playing the role $r_i$ have to exhibit.

We can then define an interaction context as a tuple $ctx \subseteq r_i \times r_j \land \forall i, j \in \text{card}(R), \forall r_i, r_j \in R$, that means that a context $ctx$ is a set of associations between two roles belonging to the set of roles $R$. Please note that the context can be smaller than the whole set $R$ due to possible constraints or incompatibilities between roles. For instance, it is possible to deny the association between the manager role and the postman one, since the latter should be associated with the former indirectly via an agent playing the employee role.

Now the definition of a role assumption is quite straightforward, since it can be though as a tuple $p = <r_0, \alpha r_0 > : r_0 \in R$ which means that the agent $\alpha r_0$ has assumed the role $r_0$.

A role $r_j$ represents an evolution of the role $r_i$ if $Q_j \subseteq Q_i$, which means that an agent playing the role $r_j$ has more starting-requirements of $r_i$ and, in other words, before the agent can assume the role $r_j$ it must own all requirements of the role $r_i$ and those specific of the role $r_j$. Please note that if the two roles $r_i$ and $r_j$ belong to the same context $ctx$, then $r_j$ is an evolution of $r_i$ if $Q_j \subseteq Q_i$. 
5 An Application Example

In order to show how roles can be useful in coordination among MAS applications, this section reports an application example that is implemented in the BRAIN framework dealing with the concepts and definitions exposed above.

The application presented here is thought to support e-democracy [1, 4, 10], a quite new scenario whose aim is to help citizens performing democratic actions (such as vote, attending a political meeting, paying taxes, etc.) through information technology. In particular, we present an application for e-democracy based on agents and roles, which enables citizens to attend a political meeting, to listen to candidates and to vote for one of them. In such application each user is represented by its digital counterpart: an agent playing a specific role (called user agent henceforth). User agents attend the meeting executing in an agent platform and playing an appropriate role chosen among participant, candidate, voter and chair. The chair has to manage the meeting, granting the speech to only one candidate per time, the participants attend the meeting listening to the candidates and, in the case, becoming candidates themselves. Finally, the voters express their preference related to one candidate through an e-vote.

Figure 8 shows a possible scenario of the application. Please note that there are no agents playing the voter role since it will be available only after each candidate as performed its speech (i.e., the meeting is ending). In the following we will show how the application works by defining the characteristics of each role.

First of all user agents can attend the meeting only if they represent citizens. User agents can then become candidates only if they are participants that represent a political party and has a speech related to the meeting subject. A user agent can play the voter role only if it has a vote certificate. The last role, chair, can be played if the user agent has already attended other meetings and knows very well the current subject, that means has experienced conventions.

Starting from the above considerations it is possible to identify the candidate role as an evolution of the participant one, while the chair role is not, since playing such role does not need starting-requirements common to the candidate/participant roles. Finally, the voter role has a common starting-requirement with the participant one, that is the needing to be played by a citizen, but adds another one that is the need to be old enough to vote. Also the voter role is an evolution of the participant one.

Using the formal notation introduced in the section 4.3 we can represent the application scenario as follows:

\[
R = \{\text{chair, participant, candidate, voter}\} \\
ctx_1 = \{\text{chair} \times \text{participant} \times \text{candidate}\} \\
ctx_2 = \{\text{chair} \times \text{voter}\} \\
Q_{\text{participant}} = \{\text{is a citizen}\} \\
Q_{\text{candidate}} = \{\text{is a citizen, represent a political party, has a speech}\} \\
Q_{\text{chair}} = \{\text{participated other meetings, knows the subject}\} \\
Q_{\text{voter}} = \{\text{is a citizen, enough old to vote}\} \\
B_{\text{chair}} = \{\text{define the beginning and the end of a speech, define the beginning and the end of the vote process}\} \\
B_{\text{voter}} = \{\text{receive a preference from the user}\} \\
\]

Note that there are two separated contexts, \(ctx_1\) and \(ctx_2\), since the application can be separated into two different phases dominated by (partially) different roles.

As already written, the candidate role represents an evolution of the participant, that means that the candidate is an enhanced participant. Nevertheless this does not mean that the agent must become firstly a participant and then a candidate, since the assumption is driven by starting-requirements.

The use of roles in this kind of application enables agents to dynamically change their behaviour and capabilities. Furthermore it is possible to embed in roles the coordination rules, so that agents can use transparently roles disregarding about such details. Moreover, it is simply possible to change role implementations in order to change the effective application behaviour, and this lead to a more flexible coordination architecture.

6 Related Work

At best of our knowledge there are no other role approaches that propose a concept of role evolution as we do.

TRANS [12] introduces the concept of priority, which permits to choose among role assumptions ordered by the priority level. Even if this can be thought as a kind of evolution, since the priority defines the assumption chain, it is not flexible as our approach. In fact, the sole concept of priority does not allow developers to strictly define requirements to assume a role. This can lead to a misbehavior, especially in the case two roles have the same priority, while our approach permits to distinguish
among them simply defining more requirements.

Fasi’s approach [11], introduces a stricter notation than our approach, but, even if it defines rights and duties related to roles, it does not take into consideration a real role evolution and does not support the implementation phase, as BRAIN does.

Other role approaches, such as GAIA [22], RoleEP [20], Truce [15] mainly focuses on interactions among agents modeled as roles, without concentrating in the experience that roles can grant to the agents allowing them to evolve.

Conclusions

In this paper we have proposed a new definition of the concept of role, based on the needing of requirements, in order to better simulate the real life in the digital world of agents. Thanks to this we believe it is possible to better develop agent-based applications where agents can really act on behalf of their users, since they can follow similar rules.

Thanks to this definition we can model the evolution of an agent in a simple way. Nevertheless this can lead to the need of common values to express requirements, since the evolution can involve roles belonging to different contexts. This represents work in progress about the presented new definition of the concept of role.

We have proposed a formal notation to define role characteristics, interaction contexts and assumption process. This notation can help developers analyzing and understanding role based applications and the dependencies in them.

Currently we are applying the above new definition to the BRAIN framework, which we have already used to develop several MAS application based on roles.

References


