Merging Roles in Coordination and in Agent Deliberation

Guido Boella[1], Valerio Genovese[1], Roberto Grenna[1], and Leendert van der Torre[2]

[1] Dipartimento di Informatica, Università di Torino,
guido@di.unito.it, valerio.click@gmail.com, grenna@di.unito.it
[2] University of Luxembourg, Luxembourg, leendert@vandertorre.com

Abstract. In this paper we generalize and merge two models of roles used in multiagent systems which address complementary aspects: enacting roles and communication among roles in an organization or institution. We do this by proposing a metamodel of roles and specializing the metamodel to fit two existing models. We show how the two approaches can be integrated since they deal with complementary aspects: Boella [1] focuses on roles as a way to specify interactions among agents, and, thus, it emphasizes the public character of roles. Dastani [2] focuses instead on how roles are played, and thus it emphasizes the private aspects of roles: how the beliefs and goals of the roles become the beliefs and goals of the agents. The former approach focuses on the dynamics of roles in function of the communication process. The latter focuses on agents' internal dynamics when they start playing a role or shift the role they are currently playing.

keywords: Roles, Organizations, O.O. Modeling, MAS, Security.

1 Introduction

In the last years, the usefulness of roles in designing agent organizations has been widely acknowledged. Many different models have been designed. Some of them use roles only in the design phases of a MAS [3], while other ones consider roles as first class entities which exist also during the runtime of the system [4]. There are approaches that underline how roles are played by agents [2], other ones on how roles are used in communication among agents in organizations [1]. This heterogeneity of the way roles are defined and used in MAS risks to be a danger for the interoperability of agents in open systems, since each agent entering a MAS can have a radically different notion of role. Thus, the newly entered agents cannot be governed by means of organizations regulating the MAS. Imposing to all agent designers a single notion of role is a strategy that cannot have success. Rather, it would be helpful to design both multiagent infrastructures that are able to deal with different notions of roles, and to have agents which are able to adapt to open systems which use different notions of roles in organizations. This alternative strategy can be costly if it is not possible to have a general model of role that is compatible, or can be made compatible with other existing concepts.

In this paper we generalize and merge two models of roles used in multiagent systems, in order to promote the interoperability of systems. The research question is: How

to combine the model of role enactment by Dastani [2] with the model of communication among organizational roles of Boella [1]?

We answer these questions by extending to agents a metamodel of roles developed for object oriented systems [5]. The relevant questions, in this case, are: how to introduce beliefs, goals and other mental attitudes in objects, and how to pass from the method invocation paradigm to the message passing paradigm.

Then we specialize the metamodel to model two existing approaches and we show how they can be integrated in the metamodel since they deal with complementary aspects. We choose to model the proposals of Boella [1] and Dastani [2] since they are representative of two main traditions. The first tradition is using roles to model the interaction among agents in organizations, and the second one is about role enactment, i.e., to study how agents have to behave when they play a role.

From one side, organizational models are motivated by the fact that agents playing roles may change, for example a secretary may be replaced by another one if she is ill. Therefore, these models define interaction in terms of roles rather than agents. In Boella [1] roles model the public image that agents build during the interaction with other agents; such image represents the behavior agents are publicly committed to. However, this model leaves unspecified, how given a role, its player will behave. This is a general problem of organizational models which neglect that when, for example, a secretary falls ill, there are usually some problems with ongoing issues (the new secretary does not know precisely the thing to be done, arrangements already made etc.). So having a model of enacting and deacting agents surely leads to some new challenges, which could not be discussed, simulated or formally analyzed without this model.

In contrast, the organizational view focuses on the dynamics of roles in function of the communication process: roles evolve according to the speech acts of the interactants, e.g. the commitment made by a speaker or the commands made by other agents playing roles which are empowered to give orders. In this model roles are modeled as sets of beliefs and goals which are the description of the expected behavior of the agent. Roles are not isolated, but belongs to institutions, where constitutive rules specify how roles change according to the moves played in the interactions by the agents enacting them.

Dastani [2] focuses, instead, on how roles are played by an agent, and, thus, on the private aspects of roles. Given a role described in terms of beliefs, goals, and other components, like plans, the model describes how these mental attitudes become the beliefs and goals of the agents. In this approach roles are fixed descriptions, so they do not have a dynamics like in the model of [1]. Moreover, when roles are considered inside organizations new problems for role enactment emerge: for example, how to coordinate with the other agents knowing what they are expected to do in their role, and how to use the powers which are put at disposal of the player of the role in the organization. The same role definition should lead to different behaviors when the role is played in different organizations.

In contrast, it specifies the internal dynamics of the agents when they start playing (or enacting in their terminology) a role or shift the role they are currently playing (called the activated role). So they model *role enacting agents*: agents that know which roles they play, the definitions of those roles, and which autonomously adapt their mental states to play the roles.

Despite the apparent differences, the two approaches are compatible since they both attributes beliefs and goals to roles. So we study by means of the metamodel how they can be combined to have a more comprehensive model of roles.

The paper is structured as follows. In Section 2 we describe the requirements on agents and roles in order to build a metamodel; in Section 3 we formally define the metamodel for roles together with its dynamics; in Section 4 we define the basic notions to model agents that play roles; Section 5 deals with the modeling of enacting agents as in Dastani [2]; Section 6 introduces and models roles to deal with coordination in organizations; in Section 7 we merge Dastani [2] and Boella [1] into the framework introduced in Section 3; Conclusions end the paper.

2 Agents and roles

Since the aim of this paper is to build a metamodel to promote interoperability, we make minimal assumptions on agents and roles.

The starting point of our proposal is a role metamodel for object orientation. The relation of objects and agents is not clear, and to pass from object to agents we take inspiration from the Jade model [6]. Agents, differently than objects, do not have methods that can be invoked starting from a reference to the object. Rather, they have an identity and they interact via messages. Messages are delivered by the MAS infrastructure, so that agents can be located in different platforms. The messages are modeled via the usual send-receive protocol. We abstract in the metamodel from the details of the communication infrastructure (whether it uses message buffers, etc.). Agents have beliefs and goals. Goals are modeled as methods which can be executed only by the agent itself when it decides to achieve the goal. As said above, we propose a very simple model of agents to avoid controversial issues. When we pass to roles, however, controversial issues cannot be avoided. The requirements to cope with both models of roles we want to integrate are:

- Roles are instances, associated in some way to their players.
- Roles are described (at least) in terms of beliefs and goals.
- Roles change over time.
- Roles belong to institutions, where the interaction among roles is specified.
- The interaction among roles specifies how the state of roles changes over time.

In Boella [1] roles are used to model interactions, so agents exchange messages according to some protocol passing via their roles. This means that the agent have to act on the roles, e.g., to specify which is the move the role has to play in certain moment. Moreover, roles interact with each other. Dastani's [2] model specifies how the state of the agent changes in function of the beliefs and goals of the roles it plays. However, it does not consider the possibility that the state of the role change and, thus, it ignores how the agent becomes aware of the changes of beliefs and goals of the role.

To combine the two models we have to specify how the interaction between an agent and its role happens when the agent changes the state of the role or the state of the role is changed by some event. A role could be considered as an object, and its player could invoke a method of the role. However, this scenario is not possible, since

the roles are strictly related to the institution they belong to, and we cannot assume that the institution and all the agents playing roles in the institution are located on the same agent platform. So method invocation is not possible unless some sophisticated remote method invocation infrastructure is used. Moreover, the role has to communicate with its player when its beliefs and goals are updated. Given that the agent is not an object, the only way is that a role sends a message to its player. As a consequence, we decide to model the interaction between the agent and the role by means of messages too.

Finally, we have to model the interaction among roles. Since all roles of an institution belong to the same agent platform, they do not necessarily have to communicate via messages. To simplify the interaction, we model communication among roles by means of method invocation.

The fact that roles belong to an institution has another consequence. According to the powerJava [7] model of roles in object oriented programming languages, roles, seen as objects, belong to the same namespace of the institution. This means that each role can access the state of the institution and of the sibling roles. This allows to see roles as a way to specify coordination. In a sense, roles are seen both as objects, from the internal point of view of the institution they belong to, and as agents, from the point of view of their players, with beliefs and goals, but not autonomous. Their behavior is simply to: (1) receive the messages of their players, (2) execute the requests of their player of performing the interaction moves according to the protocol allowed by the institution in that role, (3) send a message to their players when the interaction move performed by the role itself or by some other role results in a change of state of the role.

3 A Logical Model for Roles

In Genovese [5] the model is structured in three main levels: universal, individual and dynamic; here we decide not to talk about the universal level and concentrate ourself on agents dynamics. We define the formalism of the framework in a way as much general as possible, this gives us an unconstrained model where special constraints are added later.

3.1 Individual level

This level is composed by a *snapshot model* that describes in a particular moment the relationships between individual players contexts and roles, and a dynamic model which links snapshots and actions modeling how the system changes when an action is executed. In the formalization of the model we use *objects* as basic elements upon which the model is based.

Definition 1 A *snapshot model* is a tuple

$$< \mathsf{O}, \mathsf{R\_types}, \mathsf{I\_contexts}, \mathsf{I\_players}, \mathsf{I\_roles}, \mathsf{Val}, \mathsf{I\_contraints} \\ \mathsf{I}_{\mathsf{Roles}}, \mathsf{I\_Attributes}, \mathsf{I\_Operations}, \mathsf{I}_{\mathsf{Attr}} >$$

– O is a *domain* of objects, for each object o is possible to refer to its attributes and operations through $\pi_{I\_Attr}(o)$ and $\pi_{I\_Op}(o)$, respectively.

– R_types is a set of types of roles.
– I_contexts $\subseteq$ O is a set of institutions (referred as *institutions*).
– I_players $\subseteq$ O is a set of actors (referred as *actors*).
– I_roles $\subset$ O is a set of *roles instances* (referred as *roles_instances*).
– I_Attributes is the set of attributes.
– I_Operations is the set of operations.
– Val is a set of *values*.
– I_constraints is a set of integrity rules that constraint elements in the snapshot.

The snapshot model has also a few functions:

– I_{Roles} is a *role assignment function* that assigns to each role R a relation on I_context x I_players x I_roles.
– I_{Attr} is an *assignment function* which it takes as arguments an object $d \in$ O, and an attribute p $\in \pi_{I\_Attr}(d)$, if p has a value $v \in$ Val it returns it, $\emptyset$ otherwise.

Generally, when a role instance x is an individual of the type D, we write $x :: D$. If $a \in \pi_{I\_Attr}(x)$ we write x.a $\in$ I_Attributes as the attribute instance assigned to object x, the same holds for elements in I_Operations. (i,a,o)$\in I_{Roles}(R)$ means: "the object o represents agent a playing the role R in institution i", often written *R(i,a,o)*, and o is the *role instance*.

3.2 The dynamic model

The dynamic model relies on the individual level and defines a structure to properly describe how the framework evolves as a consequence of executing an action on a snapshot. In Section 4 and 5, we describe how this model constraints agents' dynamics.

Definition 2 A *dynamic model* is the following tuple

$$< \mathsf{S}, \mathsf{TM}, \mathsf{Actions}, \mathsf{Requirements}, \mathsf{D\_constraints}, \mathsf{I}_{\mathsf{Actions}}, \mathsf{I}_{\mathsf{Roles_t}}\ \pi_{\mathsf{Req}}, \mathsf{I}_{\mathsf{Requirements_t}} >$$

– S is a set of *snapshots*.
– TM $\subseteq$ S x $\mathbb{N}$: it is a time assignment relationship, such that each snapshot has an associated unique time t.
– Actions is a set of actions.
– Requirements is a set of requirements for playing roles in the dynamic model.
– D_contraints is a set of integrity rules that constraints the dynamic model.
– $\mathsf{I}_{\mathsf{Actions}}$ maps each action from Actions to a relation on a set of snapshots P. $I_{Actions}(s,a,t)$ tells us which snapshots are the result of executing action a at time t from a certain snapshop. [1] This function returns a couple in TM that binds the resulting snapshot with time $t+1$. In general, to express that at time t is carried action a we write a_t.

[1] Notice that given an action, we can have several snapshots because we model actions with modal logic in which, from a world it is possible to go to more than one other possible world. This property is often formalized through the *accessibility relationship*. Thus, each snapshot can be seen as a possible world in modal logic.

– About I_{Roles_t}, $R_t(i,a,o)$ is true if there exists, at a time t, the *role instance* $R(i,a,o)$.
– $\pi_{Req}(t,R)$ returns a subset of Requirements present at a given time t for the role of type R, which are the requirements that must be fulfilled in order to play it.
– $I_{Requirements_t}$ is a function that, given (i,a,R,t) returns True if the actor a fills the requirement in $\pi_{Req}(t,R)$ to play the role R in the institution i, False otherwise. We often write $Req_t(i,a,R)$.

Intuitively, the snapshots in S represent the state of a system at a certain time. Looking at I_{Actions} is possible to identify the *course* of actions as an ordered sequence of actions such that $a_1;b_2;c_3$ represents a system that evolves due to the execution of a, b and c at consecutive times. We refer to a particular snapshot using the time t as a reference, so that for instance $\pi_{I\_Attr_t}$ refers to π_{I_Attr} in the snapshot associated with t in TM.

Actions are described using dynamic modal logic [8], in paricular they are modelled through *precondition laws* and *action laws* of the following form:

$$\Box(A \wedge B \wedge C \supset \langle d \rangle \top) \tag{1}$$

$$\Box(A^{'} \wedge B^{'} \wedge C^{'} \supset [d]E) \tag{2}$$

Where the $\Box$ operator expresses that the quantified formulas holds in all the possible words. *Precondition law* (1) specifies the conditions A,B and C that make an atomic action d executable in a state. (2) is an *action law*[2] which states that if preconditions $A^{'}$,$B^{'}$ and $C^{'}$ to action d holds, after the execution of d also E holds.

In addition we introduce *complex actions* which specify complex behaviors by means of *procedure definitions*, built upon other actions. Formally a complex action has the following form: $\langle p_0 \rangle \varphi \subset \langle p_1; p_2 ....; p_m \rangle \varphi$; p_0 is a *procedure name*, ";" is the *sequencing operator* of dynamic logic, and the p_i's, $i \in [1,m]$, are procedure names, atomic actions, or test actions[3].

Now we show some examples of actions that can be introduced in the dynamic model in order to specialize the model.

Role addition and deletion

For role addition and deletion actions we use, respectively $R,i \hookrightarrow_t a$, and $R,i \hookleftarrow_t a$. Then using the notation of dynamic logic introduced above, we write: $\Box(Req_t(i,a,R) \supset \langle R,i \hookrightarrow_t a \rangle \top)$ to express that, if actor a fills the requirements at time t ($Req_t(i,a,R)$ is True), a can execute the role addition action that let him play role of type R.

The above definition gives us the possibility to model that a role assignment introduces a role instance: $\Box(\top \supset [R,i \hookrightarrow_t a]\exists x R_{t+1}(i,a,x))$, or the fact that if a does not already play the role R within institution i, then the role assignment introduces exactly one role instance: $\Box(\neg\exists x R(i,a,x) \supset [R,i \hookrightarrow_t a]\exists! x R_{t+1}(i,a,x))$

[2] Sometimes action laws are called *effect rules* because E can be considered the effect of the execution of d.

[3] Test actions are of the form $\langle \psi? \rangle \varphi \equiv \psi \wedge \varphi$.

Methods

There are other actions through which is possible to change the model as well, for instance agents may assign new values to their attributes [5].

Here, we will focus on the case in which the attribute's values can be changed by the *objects themselves*. What we will do is to define *methods* of objects with which they can change attributes of their own or those of others. Actually, to simplify the model, we define one single primitive action: $\mathsf{set}_t(o_1, o_2, \mathsf{attr}, v)$, which means that object o_1 sets the value of attr on object o_2 to v at time t. If o_1 and o_2 are autonomous agents, the $\mathsf{set}(o_1, o_2, \mathsf{attr}, v)$ can be executed only when $o_1 = o_2$.

Now, we will of course have that: $\Box(\top \supset [\mathsf{set}_t(o_1, o_2, \mathsf{attr}, v)]\mathsf{attr}_{t+1}(o_2) = v)$, which means that in any state, after the execution of set, if the action of setting this attribute succeeds, then the relevant object will indeed have this value for that attribute.

Operations

Elements of our framework come with *operations* that can be executed at the individual level in order to change the model dynamically, the semantics of each operations can be given exploiting the actions defined for the dynamic model. Suppose, for instance, to have an object individual $x :: Person$ with x.mail_address attribute, and an operation x.change_mail that changes the value of x.mail_address to its argument. Using the set primitive is possible to define how the model evolves after the execution of x.change_mail operation trough the following axiom:

$$[\mathsf{x.change\_mail}_t(s)]\varphi \equiv [\mathsf{set}_t(x, x, \mathsf{mail\_adress}, s)]\varphi$$

, where $\mathsf{x.change\_mail}_t(s)$ identifies the action carried by x at time t to execute the instance operation x.change_mail; objects can execute only operations that are assigned to them by I_OS relation. In Section 5 we define exec of certain operations as complex actions because we have to describe a more complex semantics.

4 Enact and Deact Roles

In Dastani [2], the problem of formally defining the dynamics of roles, is tackled identifying the actions that can be done in a *open system* such that agents can enter and leave. Here, four operations to deal with role dynamics are defined: *enact* and *deact*, which mean that an agent starts and finishes to occupy (play) a role in a system, and *activate* and *deactivate*, which means that an agent starts executing actions (operations) belonging to the role and suspends the execution of the actions. Although is possible to have an agent with multiple roles enacted simultaneously, only one role can be *active* at the same time. Before diving into modeling the four basic operations to deal with roles, we need to match our framework with a few concepts defined in [2], following we report a list of elements together with their definition and then how they fit in our meta-model:

– *Multiagent system*: In [2] roles are taken into account at the implementation level of *open MAS*, they belong to the system which can be entered or left by agents

dynamically. In our framework is possible to view a system as a context to which are linked all roles that can be played by the agents.

- *Agent role*: A role is a tuple $\langle \sigma, \gamma, \omega \rangle$. Where σ are beliefs, γ goals and ω rules representing conditional norms and obligations. This definition specifies a role "in terms of the information that becomes available to agents when they enact the role, the objectives or responsibilities that the enacting agent should achieve or satisfy, and normative rules which can for example be used to handle these objectives" [2]. With this view we define, for *roles* of our framework, a set of complex attributes $\{\mathsf{beliefs}, \mathsf{goals}, \mathsf{plans}, \mathsf{rules}\} \in \mathsf{I\_Attr}$ togheter with the *operations* that represent actions that an agent can carry out when it *activates* the roles instance choosing it from the set of roles it is playing.
- *Agent type*: We consider an agent type "as a set of agent roles with certain constraints and assume that an agent of a certain type decides itself to enact or deact a role". To talk about agent types we use *classes* introduced in the framework as a specification of agent instances at the individual level, with this in mind we use the PL relationship to link *agent classes* to *agent roles* (role's classes) so that the set of roles that an agent can enact (play), is constrainted by I_PL.
- *Role enacting agent*: "We assume that role enacting agents have their own mental attitudes consisting of beliefs, goals, plans, and rules that may specify their conditional mental attitudes as well as how to modify their mental attitudes. Therefore, role enacting agents have distinct objectives and rules associated to the active role it is enacting, and sets of distinct objectives and rules adopted from enacted but inactive roles". In our framework we define a *role enacting agent* as a instance x having a set of attributes A that represent the internal structures used to deliberate.

$$A = \{\mathsf{beliefs_a}, \mathsf{objectives_a}, \mathsf{plans_a}, \mathsf{rules_a}, \mathsf{enacted\_roles}[], \mathsf{active\_role}\} \in \pi_{I\_Attr}(x)$$

The enacted_roles attribute is a role ordered record where each entry with index i corresponds to a triple $\langle \sigma_i, \gamma_i, \omega_i \rangle$ which represents the set of beliefs, objectives, plans and rules associated to roles instance i enacted by x.

As introduced above, the model in [2] identifies four operations to deal with role dynamics, in order to to grasp the fundamental ideas proposed in the cited paper, we redefine the *enact*, *deact*, *activate* and *deactivate* operations respecting their original meaning. Given an agent x, a role instance $i :: R$ played by x in context c s. t.,

$$\begin{gathered}\{\mathsf{beliefs_r}, \mathsf{objectives_r}, \mathsf{plans_r}, \mathsf{rules_r}\} \in \pi_{I\_Attr}(i) \\ \{\mathsf{beliefs_a}, \mathsf{objectives_a}, \mathsf{plans_a}, \mathsf{rules_a}, \mathsf{enacted\_roles}[], \mathsf{active\_role}\} \in_{SA} \pi_{I\_Attr}(x) \\ \{\mathsf{enact}, \mathsf{deact}, \mathsf{activate}, \mathsf{deactivate}\} \in \pi_{I\_Op}(x)\end{gathered}$$

Next we report the semantics of each operation exploiting the set primitive:

$$\begin{aligned}\langle \mathsf{x.enact_t(i)} \rangle \varphi \subset \langle \mathsf{R}, \mathsf{s} \hookrightarrow \mathsf{x}; \mathsf{set_t}(\mathsf{x}, \mathsf{x}, \mathsf{beliefs_a}, \mathsf{beliefs_a} \cup \mathsf{beliefs_r}); \\ \mathsf{set_t}(\mathsf{x}, \mathsf{x}, \mathsf{enacted\_roles[i]}, < \mathsf{objectives_r}, \mathsf{plans_r}, \mathsf{rules_r} >) \rangle \varphi\end{aligned} \tag{3}$$

$$\langle \mathsf{x.deact_t(i)} \rangle \varphi \subset \langle \mathsf{R}, \mathsf{s} \hookleftarrow \mathsf{x}; \mathsf{set_t}(\mathsf{x}, \mathsf{x}, \mathsf{enacted\_roles[i]}, \mathsf{null}) \rangle \varphi \tag{4}$$

$$\langle \mathsf{x.activate_t(i)} \rangle \varphi \subset \langle \mathsf{set_t}(\mathsf{x}, \mathsf{x}, \mathsf{active\_role}, \mathsf{enacted\_roles[i]}) \rangle \varphi \tag{5}$$

$$\langle \mathsf{x.deactivate_t(i)} \rangle \varphi \subset \langle \mathsf{set_t}(\mathsf{x}, \mathsf{x}, \mathsf{active\_role}, \mathsf{null}) \rangle \varphi \tag{6}$$

5 The public dimension of roles

In Boella-Van der Torre [9] roles are introduced inside institutions to model the interaction among agents. In Boella [1] the model is specifically used to provide a semantics for agent communication languages in terms of public mental attitudes attributed to roles.

The basic ideas of the model are: (1) roles are instances with associated beliefs and goals attributed to them. These mental attitudes are public. (2) The public beliefs and goals attributed to roles are changed by speech acts executed either by the role or by other roles. The former case accounts for the addition of preconditions and of the intention to achieve the rational effect of a speech act, the latter one for the case of commands or other speech acts presupposing a hierarchy of authority among roles. (3) The agents execute speech acts via their roles.

In order to maintain the model simple enough, we model message passing extending the dynamic model with two actions (methods) $\mathsf{send(x,y,sp)}$ and $\mathsf{receive(y,x,sp)}$. Where $\mathsf{send(x,y,sp)}$ should be read as the action carried by x of sending a speech act (sp) to y and $\mathsf{receive(y,x,sp)}$ is the complementary action of y receiving the message from x. It must be underlined that arguments x and y can be agents or roles. A role only listens for the messages sent by the agents playing it:

$$\langle \mathsf{listen(r)} \rangle \varphi \subset \langle \mathsf{P}; \mathsf{played\_by(r,x)?}; \mathsf{receive(r,x,sp)}; \mathsf{D} \rangle \varphi$$

These rules define a *pattern* of protocol where P and D have to be read as possible other actions that can be executed before and after the receive. The reception of a message from the agent has the effect of changing the state of other roles. For example, a command given via a role amounts to the creation of a goal on the receiver if the sender has authority (within the system) over it.

$\Box(\mathit{authority}_{sys}(r, \mathsf{request} \supset [\mathsf{receive(r,x,request(r,r',act))}])\mathbf{G}^{\mathsf{r}'}_{\mathsf{t}}(\mathsf{act}))$[4]
To produce a speech act, the agent has to send a message to the role specifying the illocutive force, the receiver and the content of the speech act:

$$\langle \mathsf{communicate(a)} \rangle \varphi \subset \langle \mathsf{P}; \mathsf{send(x,r,sp)}; \mathsf{D} \rangle \varphi$$

6 The combined model

The two models presented above model complementary aspects of roles: the public character of roles in communication and how agents privately adapt their mental attitudes to the roles they play.

In this section we try to merge the two approaches using the metamodel we presented. On the one hand, Boella's model [1] is extended from the public side to the private side, by using Dastani [2] as a model of role enacting. In this way, the expectations described by the roles resulting from the interaction among agents can become

[4] $\mathsf{request(r,r',act)}$ means: role r asks to r''s player to do act. $\mathit{authority}_{sys}(r, \mathsf{request})$ expresses that role r has the authority to make a $\mathsf{request}$ within system *sys*.

a behavior of agents and they do not remain only a description. On the other hand, Dastani's model [2] is made more dynamic. In the original model the role is given as a fixed structure. The goals of agent can evolve according to the goal generation rules contained in it, but the beliefs and goals described by the role cannot change. This is unrealistic, since during the activity of the agent enacting its role, it is possible that further information are put at disposal of the role and that new responsibilities are assigned, etc.

In order to merge the two models within the same framework, we need to add (complex) actions which are able to grasp the dynamics introduced in [1] and [2]. Interactions among agents is done through message passing and, in particular, through actions send and receive introduced in section 6. Next we are going to introduce all the speech-acts and complex actions which are needed to grasp the combined model and then we introduce a running example to clarify their use defining a *course* of actions in the dynamic model defined in section 3.2. An agent who wants to play a role within an *open system* has to ask to the system for a role instance; this process is handled by two speech act: $\mathsf{ask\_to\_play(R)}$ and $\mathsf{accept\_to\_play(r,A)}$, where the first one is sent from the agent to the system in order to ask to play a role of type R, whereas the second is sent from the system to the agent, together with the identifier of the role instance r and a set A of other role instances present in the system, in order to inform the agent with which roles is possible to interact. Next we report the two *effect rules* associated:

$$\Box(\top \supset [\mathsf{receive(s,x,ask\_to\_play(R)); send(s,x,accept\_to\_play(r,A)}] \\ \mathsf{played\_by_{sys}(r,x,s)} \tag{7}$$

$$\Box(\top \supset [\mathsf{send(x,s,ask\_to\_play(R)); receive(x,s,accept\_to\_play(r,A)}] \\ \mathsf{played\_by_{ag}(r,x,s))} \tag{8}$$

Where s is the system, x the agent, and r a role instance of type R. In this section we use x,y,z... to denote agents, s for the system and $r, r', r'' \ldots$ for role instances. Notice that $\mathsf{played\_by_{sys}(r,x,s)}$ and $\mathsf{played\_by_{ag}(r,x,s)}$ refer to two different infrastructures; in Rule 7 is the system that, after having acknowledged the agent request, knows that x is going to play r, whereas in Rule 8 is the agent that becomes aware of the play relation between x and r. To link the two predicates with the logical model introduced in Section 3 we have that: $\mathsf{played\_by_{sys}(r,x,s)} \wedge \mathsf{played\_by_{ag}(r,x,s)} \rightarrow \mathsf{R(s,x,r)}$. When we are dealing with a single system we can omit s writing $\mathsf{played\_by_{sys}(r,x)}$ and $\mathsf{played\_by_{ag}(r,x)}$. To enact a role, an agent, provided the identifier of the role instance it wants to enact, has to send a message to the role and to wait till the role replies with the information about the state of the role: its beliefs, goal, plans, etc. When the state is received, the agent can enact the role in the same way described by Rule 3 in Section 5. In order to model such interaction we introduce two complex actions tell_enact, accept_enact and two speech acts accept_enact and inform_enact.

$$\langle \mathsf{tell\_enact(x,r)} \rangle \varphi \subset \langle \mathsf{played\_by_{ag}(r,x)?; (send(a1,r1,enact(x,r))} \rangle \varphi \tag{9}$$

$$\langle \mathsf{accept\_enactment(r,x)} \rangle \varphi \subset \langle \mathsf{receive(r,x,enact(x,r)); played\_by_{sys}(r,x)?;} \\ \mathsf{send(r,x,inform\_enact(<beliefs_r, objectives_r, plans_r, rules_r>))} \rangle \varphi \tag{10}$$

When the agent receives the specification of the role he wishes to enact, it can internalize them as in Rule 3:

$$\Box(\top \supset [\mathsf{receive}(x,r,\mathsf{inform\_enact}(< \mathsf{beliefs}_r, \mathsf{objectives}_r, \mathsf{plans}_r, \mathsf{rules}_r >))] \\ \mathbf{B}^x(\mathsf{beliefs}_r) \wedge x.\mathsf{enacted\_roles}[r] =< \mathsf{objectives}_r, \mathsf{plans}_r, \mathsf{rules}_r >)^5 \quad (11)$$

In this combined view is possible that role's specifications change dynamically, in that case it is up to the role to send a message to its player each time its state is updated:

$$\langle \mathsf{udpate\_state}(r,x)\rangle\varphi \subset \langle \mathsf{played\_by}_{\mathsf{sys}}(r,x)?; (\neg \mathbf{G}^r_t(q) \wedge \mathbf{G}^r_{t+1}(q))?; \\ \mathsf{send}(r,x,\mathsf{inform\_goal}(q))\rangle\varphi \quad (12)$$

Last but not least, we need to model the deactment of a role respecting the formalization as in Rule 4, therefore we introduce two speech acts deact, ok_deact and a complex action confirm_deact defined as follows:

$$\langle \mathsf{confirm\_deact}(r,x)\rangle\varphi \subset \langle \mathsf{receive}(r,x,\mathsf{deact}); \mathsf{played\_by}_{\mathsf{sys}}(r,x)?; \\ \mathsf{send}(r,x,\mathsf{ok\_deact})\rangle\varphi \quad (13)$$

After sending ok_deact, the system will not consider anymore agent x as player of r:

$$\Box(\top \supset [\mathsf{confirm\_deact}(r,x)]\neg \mathsf{played\_by}_{\mathsf{sys}}(r,x) \quad (14)$$

If it is possible for the agent to deact the role, it will receive an ok_deact from its role:

$$\Box(\top \supset [\mathsf{receive}(x,r,\mathsf{ok\_deact})]x.\mathsf{enacted\_roles}[r] = \mathsf{null} \wedge \neg \mathsf{played\_by}_{\mathsf{ag}}(r,x)) \quad (15)$$

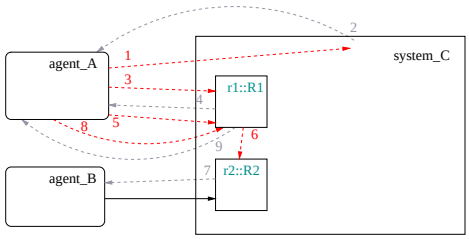

Fig. 1. Roles in MAS

Fig. 1 depicts two agents which interact through roles in an open system. At time t the system has already agent_B that enacts role r2 as represented by the black arrow which goes from agent_B to r2. Following the course of actions that describe how the system evolves:

1. At time t+1 agent_A asks to institution system_C to play a role of type R1:

$$\mathsf{send}_{t+1}(\mathsf{agent\_A}, \mathsf{system\_C}, \mathsf{ask\_to\_play}(\mathsf{R1}))$$

2. At time t+2 system_C replies to agent_A assigning to him the role instance r1:

$$\mathsf{send}_{t+2}(\mathsf{system\_C}, \mathsf{agent\_A}, \mathsf{accept\_to\_play}(\mathsf{r1}, \{\mathsf{r2}\}))$$

3. At time t+3 agent_A wants to enact (internalize) role r1: $\mathsf{tell\_enact}_{t+3}(\mathsf{agent\_A}, \mathsf{r1})$
4. At time t+4 role r1 receives the speech act from agent_A asking for enactment and accepts it, replying to agent_A with its specifications: $\mathsf{accept\_enactment}_{t+4}(\mathsf{r1}, \mathsf{agent\_A})$
5. Once that agent_A has enacted the role as in Rule 3 it decides, at time t+5, to activate it [6] and then to ask to the agent playing r2 to do an action act. In other words: $\mathsf{send}_{t+5}(\mathsf{agent\_A}, \mathsf{r1}, \mathsf{request}(\mathsf{r1}, \mathsf{r2}, \mathsf{act}))$ When r1 receives a send from agent_A asking for an act of r2, first it checks if the sender has the authority in the system to ask such an act, if so r2 acquires the goal to do act:

$$\Box(\mathit{authority}_{\mathsf{sys}}(\mathsf{r}', \mathsf{act}) \supset [\mathsf{receive}(\mathsf{r}, \mathsf{agent\_A}, \mathsf{request}(\mathsf{r}, \mathsf{r}', \mathsf{act}))]\mathbf{G}^{\mathsf{r}'}(\mathsf{act}))$$

   Is important to underline that because role internals are public to other roles in the same system, it is always possible for r1 to check or modify r2's goals. So, at time t+6 we have: $\mathsf{receive}_{t+6}(\mathsf{r1}, \mathsf{agent\_A}, \mathsf{request}(\mathsf{r1}, \mathsf{r2}, \mathsf{act}))$
6. Now that r2 has updated its internal state (i.e. its goals) it must inform its player agent_B: $\mathsf{update\_state}_{t+7}(\mathsf{r2}, \mathsf{agent\_B})$, where update_state is modeled as in Rule 12
7. At time t+8 agent_A decides to deact the role r1: $\mathsf{send}_{t+8}(\mathsf{agent\_A}, \mathsf{r1}, \mathsf{deact})$
8. Finally, at time t+9, r1 confirm the deact: $\mathsf{confirm\_deact}_{t+9}(\mathsf{r1}, \mathsf{agent\_A})$

7 Conclusions and Further Works

In this article we merged two representative role's models in MAS by introducing a metamodel taken from Genovese [5] and adapting it to agents. In particular, we added representations of typical agents' mental attitudes and a framework to deal with message passing. The model has been specialized in order to describe both public and private dimensions of roles (Boella, Dastani[1,2]). Finally, we merged the two dimensions defining a group of actions together with their semantics and we modeled a running example to show a possible course of events.

Further works point in two main directions: adapting the proposed metamodel to other roles approaches like Omicini [10], and introducing a formal proof theory of roles' actions dynamics and related aspects starting from Baldoni et al [8].

References

1. G. Boella, R. Damiano, J. Hulstijn, L. van der Torre: Acl semantics between social commitments and mental attitudes. In Procs. of Workshop on Agent Communication (2006)

[6] Activating a role means to take into account its specification during the private agent deliberation process, so there is no need to introduce a public action in the dynamic model to represent the activation of a role.

2. Dastani, M., van Riemsdijk, B., Hulstijn, J., Dignum, F., Meyer, J.J.: Enacting and deacting roles in agent programming. In: Procs. of AOSE'04, New York (2004) 189–204
3. Zambonelli, F., Jennings, N., Wooldridge, M.: Developing multiagent systems: The Gaia methodology. IEEE Transactions of Software Engineering and Methodology **12(3)** (2003) 317–370
4. Colman, A., Han, J.: Roles, players and adaptable organizations. Applied Ontology (2007)
5. Valerio Genovese: Towards a general framework for modelling roles. In: Normative Multi-agent Systems. Number 07122 in Dagstuhl Seminar Proceedings (2007)
6. Bellifemine, F., Poggi, A., Rimassa, G.: Developing multi-agent systems with a FIPA-compliant agent framework. Software - Practice And Experience **31(2)** (2001) 103–128
7. Baldoni, M., Boella, G., van der Torre, L.: Roles as a coordination construct: Introducing powerJava. ENTCS Procs. of the First International Workshop on Methods and Tools for Coordinating Concurrent, Distributed and Mobile Systems (MTCoord 2005) **150** (2006) 9–29
8. M. Baldoni, C. Baroglio, A. Martelli, V. Patti: Reasoning about interaction protocols for customizing web service selection and composition. Journal of Logic and Algebraic Programming, special issue on Web Services and Formal Methods,70(1):53-73 (2007)
9. Boella, G., van der Torre, L.: The ontological properties of social roles in multi-agent systems: Definitional dependence, powers and roles playing roles. Artificial Intelligence and Law Journal (AILaw) (2007)
10. Omicini, A., Ricci, A., Viroli, M.: An algebraic approach for modelling organisation, roles and contexts in MAS. Applicable Algebra in Engineering, Communication and Computing **16** (2005) 151–178