WDS'10 Proceedings of Contributed Papers, Part I, 19–24, 2010. ISBN 978-80-7378-139-2 © MATFYZPRESS

# Role-based Approaches to Development of Multi-Agent Systems: A Survey

O. Kazík

Charles University, Faculty of Mathematics and Physics, Prague, Czech Republic.

**Abstract.** In recent works concerning open Multi-Agent Systems (MAS), the emphasis has been laid on organizational aspects of the development of agent societies. Many models of collaboration and cooperation of agents have been proposed which allow reusability of design patterns in MAS and separation of concerns. One of these approaches is the role-based model, inspired by the importance of roles in human communities. In this paper we summarize the common features of role-based methodologies, compare frameworks and apply the concepts of role-based models to a concrete scenario in the field of Computational MAS.

## Introduction

Agent is a computer system that is situated in some environment that is capable of autonomous action in this environment in order to meet its design objectives [Weiss, 1999]. Some of its features are adaptivity to changes in the environment or collaboration with other agents. Such interacting agents join in more complex societies, Multi-Agent Systems (MAS). These groups of agents gain several advantages, as are the applications in distributed systems, delegacy of subproblems on other agents, and flexibility of the software system engineering.

Many present-day applications require open societies whose agents can join, leave them or change their position in them, and which are able to cope with dynamic environments. The importance of an interaction and cooperation aspects of agents, therefore, increases. The effort to reuse MAS patterns brings the need of separation of the interaction logic from the inner algorithmic logic of an agent. There are several approaches providing such separation and modeling MAS from the organizational perspective, among others the Tuple-Spaces, Group Computation, Activity Theory or Roles [Cabri et al., 2004a]. We will examine the latter model which is inspired by a sociological concept of a role applied in several methodologies on the analysis, design, and implementation level.

## Roles

The concept of a *role* is well established in different fields of computer science. We can find the roles in the domain of software development [Fowler, 1997], e.g. in the object-oriented programming, design patterns or computer-to-human interfaces. The roles there include groups of objects with similar common behavior. The roles are also used for restricting of the access to various sources of a system to guarantee the security, for example in the Role Based Access Control (RBAC) [Ferraiolo and Kuhn, 1992], or in the Computer Supported Cooperative Work (CSCW) [Ahmed et al., 2003].

In the field of the multi-agent systems engineering, peculiarities of agents, such as the autonomy and ability to collaborate, are emphasized. Generally speaking, a role is an abstract representation of stereotypical behavior common to different classes of agents. Moreover, it serves as an interface, through which the agents obtain their knowledge of their execution environment and affect this environment. Such representation contains a set of patterns of interactions, capabilities, and knowledge which the associated agent may utilize to achieve its goals. On the other hand, the role defines constraints, which a requesting agent has to satisfy to obtain the role, and responsibilities, for which the agent playing this role holds accountable. It serves also as a mean of definition of protocols, common interactions between agents. In most approaches the assignment between roles and agents is a general relation, i.e. an agent may handle more roles as well as a role can be embodied by different classes of agents. Moreover, the agents can change their roles dynamically.

The role-based solutions may be independent of the concrete situation in a system. This allows to design an overall organization of the multi-agent systems, represented by roles and their interactions, separately from the algorithmic issues of agents and to reuse the solutions from different application contexts. The coordination of agents is based on local conditions, especially the positions of the agent

playing some role, and so even the large MAS can be built modularly out of simple organizational structures.

## Survey of existing approaches

We will describe there briefly some of the role-based approaches. The comparison is difficult for many reasons. First, the models vary in the used terminologies and definitions of roles. In addition, the roles are not exploited in all phases of the development process. Some of the approaches define roles in the analysis phase and these are transformed to agent classes during design, whereas others implement the role-based support for applications. The most distinctive role-based approaches will be examined in their development support and their definition of roles.

The *Gaia* methodology [Wooldridge et al., 2000] is a conceptual model for multi-agent system analysis and design. The phases preceding the implementation, i.e. the analysis and design, are there well separated and key concepts in each phase are identified. The goal of the analysis phase is an elaboration of functional features of the MAS and its organization, which includes: identification of the roles played by agents in the organization, interactions between these roles controlled by the protocols, and constraints on both roles and protocols maintaining their coherency. These three domains (i.e. roles, protocols, and constraints) constitute the *role model*, *interaction model*, and *organizational rules*, which are inputs of the next phase, the design phase. The actual agent system is there defined so that it will be suitable for an easy implementation. This definition results in the following three models: the *agent model*, which determines final classes playing concrete roles; *service model*, which identify the services associated with each agent to fulfill its role; and *acquaintance model*, which represents communication between agents and follows from the interaction and agent model.

The role model's domain in Gaia is a set of the key roles in the system. The roles are defined by means of the four following attributes: *protocols*, which define the specific patterns of interaction with other roles; *activities*, i.e. tasks associated with the role that an agent carries about without interaction with another agent; *permissions*, which determine access to information resources; and finally *responsibilities*, determining expected behavior and functionality of the role.

The interaction model consists of a set of protocol definitions, the definitions of fixed patterns of interactions between roles. These protocols are described by schemata with following attributes: the *initiating role*, *responding role*, *inputs*, *outputs* (i.e. information used by initiator and provided by responder), and description of protocol's *purpose*. The organizational rules control the coherency of the system and can be useful in the context of open systems. These models lead to the final agent model, assignment of agent classes to the roles, their services required to perform these roles, and acquaintances directing the interactions between agents.

The Gaia methodology represents a role-based model of MAS engineering from the organizational perspective. The models designed by means of this methodology are independent on a choice of the final implementation. On the other side, the support for the role-based implementation is not considered and the role model is used only in the analysis phase and left in the design level. There are attempts to combine the Gaia methodology with the JADE framework [Moraitis and Spanoudakis, 2004] but the flexibility resulting from the role approach is not exploited in an application. The Gaia is suitable mainly for closed systems, where the components are known at the design time since the relation between roles and agents is firm.

The aim of the *Multiagent Systems Engineering* (MaSE) [DeLoach et al., 2001] methodology is a development of general-purpose multi-agent systems. The development is again split into analysis and design phase. The analysis phase consists of three steps: *capturing the goals*, which is done by identifying the goals and subgoals of a system in scenarios, and by structuring goals into a goal hierarchy diagram; *applying the use cases*, where the scenarios are captured with desired system behaviors and event sequences; and *refining the roles*, where a role model is established.

The result of the analysis phase is thus a set of roles responsible for achieving of the system level goals. The purpose of the design phase is a definition of concepts more suitable for implementation: agents and conversations. It progresses through four steps: *construction of agent classes*, where agent classes and their conversations are identified and documented in Agent Class Diagrams; *constructing conversations*, i.e. detailed definition of conversations model by means of finite state automata; *assembling agent classes*, i.e. defining the agents' internal architecture; and *deployment design*, where the actual configuration of the system is chosen and documented in a Deployment Diagram.

The MaSE has several weaknesses: there is no mechanism for modeling MAS interaction with the environment, designed MAS have rather fixed organization, the integration of sub-teams into a

MAS is not allowed, the protocols are decomposed into small and simple pieces, and roles are again restricted to the analysis phase. Some of these disadvantages are solved in organizational extension of the methodology, O-MaSE [DeLoach, 2006].

The *ALAADIN* framework [Ferber and Gutknecht, 1998; Ferber et al., 2004] is an organization-centered generic meta-model of multi-agent systems. It defines a general conceptual structure which is utilized in the MAS development. The framework describes MAS from an organizational perspective, instead of using terms of agents' mental states (agent-centered). This model (also called AGR) focuses on three basic concepts: agent, group, and role.

An *agent* is an active, communicating entity which plays roles and is a member of groups. The model does not describe internal structure and architecture of the agent, only its expected behavior. A *group* is a set of agents and serves as a context for these agents. A *group structure*, an abstract representation of a group, is described by a set of admissible roles which agents in the group can play, and by possible interactions between the role pairs in the group. Two agents can communicate if and only if both are in the same group. On the other hand, groups can intersect due to the agents handling two different roles in different groups. *Roles*, thus, are abstract representations of the functional position of an agent in a group and at the same time they are views of agents playing them, i.e. the way other agents recognize them. The AGR model also introduces the formal model of an organizational structure defining sets of roles (groups) and their interactions and dependencies. In order to demonstrate the potentialities of the model, the MadKit agent platform have been implemented [Ferber and Gutknecht, 1998].

The positive features of the ALAADIN model are introducing modularity of MAS consisting of simpler groups and interoperability among different implementation platforms allowed by a role view of agents. However, the roles are still tightly bound to the notion of agents.

The *BRAIN* (Behavioral Roles for Agent INteractions) [Cabri et al., 2003] framework is a multi-layer role-based approach to support MAS development process. The framework provides three components: model of interactions, XML-based notation, and interaction infrastructures.

The *interaction model* in BRAIN represents a role by a set of capabilities and an expected behavior. The role's set of capabilities is a set of available *actions* which an agent playing the role can perform in order to fulfill its task. The expected behavior is a set of *events* that an agent playing the role ought to handle. This role model is described by means of the XML-based *notation*, called XRole. The tagged XRole representation of the model is a compromise of human and machine readability, interoperability, and platform independence.

The BRAIN *interaction infrastructures* are implementations of the role model. [Cabri et al., 2004b] The interaction infrastructure allows assumption and dismission of roles by agents during their lifetime, translation of actions into events and their delivery, and control local policies of the system. Different infrastructures (RoleSystem or RoleX) can be plugged-in into the system according to the need of flexibility, security, efficiency or compactness of the system without any change of the two top layers.

The BRAIN model supports MAS development in all of its phases and introduces dynamism into the agent-role relation during the run-time. Moreover, the role model of a MAS is independent on the platform which controls sending messages. On the other hand, this is provided by rather complicated process of role registration including either communication with a central unit or Java bytecode manipulation.

There exist other approaches employing the concept of roles that are not so relevant to our work. Xu, Zhang, and Patel's [Xu et al., 2007] approach proposes another formal role-based model of open MAS specified by means of *Object-Z* formalism, consisting of the role organization, role space, and agent society. *RoleEP* (Role based Evolutionary Programming, [Ubayashi and Tamai, 2000]) is a system supporting construction of cooperative mobile agent applications, which are described by mean of four basic notions: objects, roles, agents, and environments. The *TRUCE* (Tasks and Roles in a Unified Coordination Environment, [Jamison and Lea, 1999]) framework consists of concurently interpreted scripts defining coordination between agents. Comparison between main examined methodologies in their development process support is shown in the table below.

**Table 1.** Development support comparison of role-based approaches

| Approach | Analysis Support | Design Support | Implementation Support |
|---|---|---|---|
| Gaia | Roles and interactions | Agents without roles | None |
| MaSE | Roles and system goals | Agents without roles | None |
| ALAADIN | Model AGR (Agents, Groups, Roles) | | MAS platform MadKit |
| BRAIN | XML-based notation XRole | | Interaction infrastructures RoleSystem, RoleX |

## Case Study: Computational MAS

The application area of our concern is computational intelligence, namely hybrid models. These models including combinations of artificial intelligence methods, such as neural networks, genetic algorithms (GA), and fuzzy logic controllers, have shown to be promising and extensively studied research area [Bonissone, 1997]. They have demonstrated better performance over individual methods in many real-world tasks. Their disadvantage is generally higher complexity, the need to manually set them up, and tune various parameters. Also, there are not many software packages that provide a large collection of individual computational methods, as well as the possibility to connect them into hybrid schemes in various ways.

The hybrid models are thus complex systems with a large number of components and computational methods, and with potentially unpredictable interactions between these parts. Multi-agent systems are suitable solutions in order to manage this complexity. The computational MAS contains one or more computational agents, i.e. highly encapsulated objects realizing particular computational methods, collaborating with other autonomous agents to fulfill their goals. Several models of development of hybrid intelligent systems by means of MAS have been proposed, e.g. [Zhang and Zhang, 2004] and [Neruda and Beuster, 2006].

In order to verify the abilities of role-based models we will present an example of the analysis of a computational MAS scenario. We are exploiting the conceptual framework of the AGR model. Its organization-centered perspective allowing modular and variable construction of MAS is well suited especially to more complicated configurations of computational agents.

As an example we take the computational MAS from [Neruda and Beuster, 2008]. The system consists of a Task Manager agent, Data Source agent, two computational agents (RBF neural network and Evolutionary algorithm agent), and supplementary agents. In the case of RBF network, there are unsupervised (vector quantization) and supervised (gradient, matrix inverse) learning agents. The evolutionary algorithm agent needs Fitness, Chromosome, and Tuner agents.

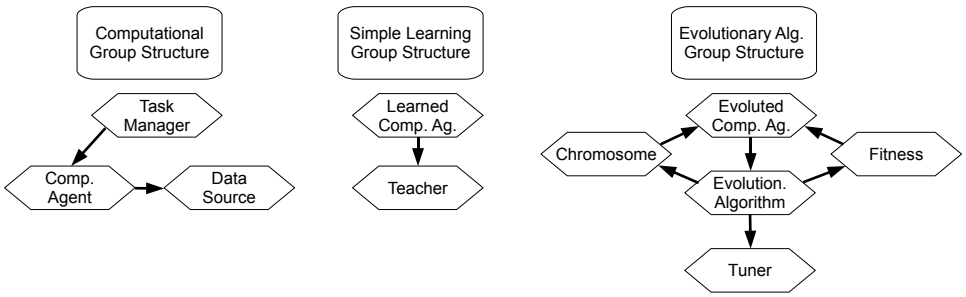


**Figure 1.** The organizational structure diagram of the computational MAS

Such a computational MAS is represented by an role organizational structure showed on the figure 1. It contains the following group structures:

- *Computational Group Structure.* It contains three roles: a Task Manager, Data Source, and Computational Agent.
- *Simple Learning Group Structure* consisting of two roles: a Teacher and Learned Computational Agent. This structure is instantiated by three groups for each Teacher (Vector Quantization, Gradient, and Matrix Inverse).
- *Evolutionary Algorithm Group Structure* which is more complicated than the previous two. It contains an Evolutionary Algorithm agent, Evolved Computational Agent, Fitness, Chromosome, and Tuner.

At the beginning of the run, only the computational group exists with the RBF network in the role of a computational agent. After the request for learning the problem by the task manager, appropriate simple learning groups are created and the learning agents are constructed/reused/found. Similarly, the evolutionary algorithm group is constructed with all supplementary agents. The interactions proceed

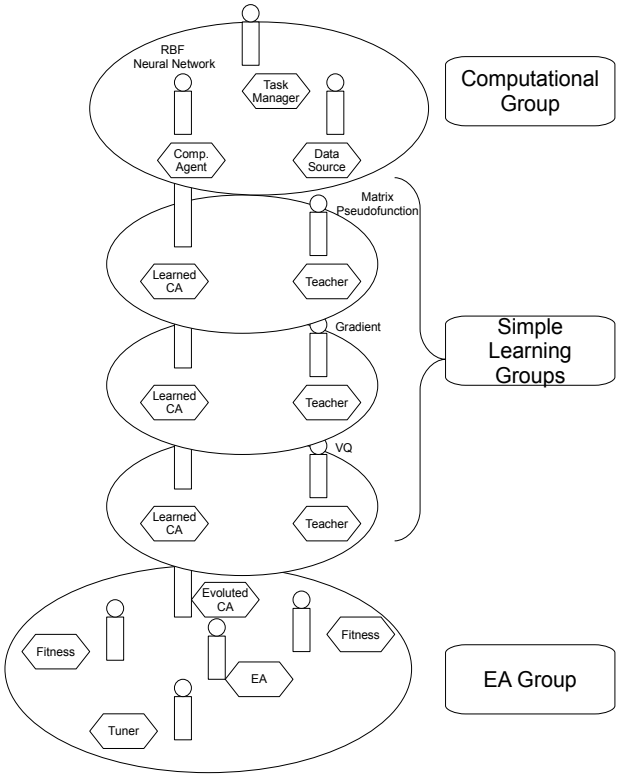


**Figure 2.** The organization of a concrete computational MAS scenario (cheeseboard notation)

according to the definition of organizational structure. The figure 2 shows an actual organization of such computational MAS.

We can see, that the role model allows to simplify the construction of more complicated computational multi-agent systems by its decomposition to the simple group structures and roles, to which the agents assigns. Moreover, the position of an agent in MAS in every moment of the run-time is defined by their roles without need to take into account their internal architecture or concrete method it implements. It also reduces the space of possible responding agents when interactions are established.

## Conclusion

In large and open multi-agent systems, where agents can join and leave different groups and behave according to its changing position in the environment, the emphasis shifts from the inner structure and algorithmic logic of individual agents to their interaction and cooperation aspects. In this paper we have examined the concept of a role which allows to design multi-agent systems from this organizational perspective.

This paper has also presented a survey of the most significant role approaches for MAS. Despite their diversity in a terminology and varying support for the phases of the development, an introduction of this concept simplifies engineering of an application. In general, a role is a concept independent of agents, to which an agent can assign, and it is defined as a set of obligations and capabilities of the agent. Protocols and groups the agents can participate are also described by means of the roles. The advantages in the development result from the separation of concerns, generality and reusability of solutions, modularity, and locality. These features significantly reduces the development process.

We have also proved usability of the role approach in computational MAS, where individual agents migrate and establish potentially unpredictable interactions. The analysis of a scenario from this field by means of concepts of agents, groups, and roles suggests possibilities of such a model.

Future research will be put in the formal ontological description of the earlier mentioned role model of computational MAS by means of Description Logics. This formalization allows to link the role-based

analysis with run-time management of the system. Maintaining consistency and matchmaking of the responding agent can be converted to a simple reasoning, integrity constraints validation or query problem in the ontological model.

**Acknowledgments.** The author would like to thank his advisor Mgr. Roman Neruda, CSc. for his advices and guidance.

## References

Ahmed, T., Kumar, R., and Tripathi, A. R., Secure management of distributed collaboration systems, Tech. rep., Dept. of Computer and Information Science, University of Minnesota, http://www.cs.umn.edu/Ajanta/papers/secure-mgmt.pdf, 2003.

Bonissone, P., Soft computing: the convergence of emerging reasoning technologies, *Soft Computing*, *1*, 6–18, 1997.

Cabri, G., Ferrari, L., and Leonardi, L., Brain: A framework for flexible role-based interactions in multiagent systems, in *Proceedings of the 2003 Conference on Cooperative Information Systems (CoopIS)*, 2003.

Cabri, G., Ferrari, L., and Leonardi, L., Agent role-based collaboration and coordination: a survey about existing approaches, in *Proc. of the Man and Cybernetics Conf.*, 2004a.

Cabri, G., Ferrari, L., and Zambonelli, F., Role-based approaches for engineering interactions in large-scale multi-agent systems, in *Proceedings of the SELMAN 2004*, edited by C. L. et al., pp. 243–263, 2004b.

DeLoach, S. A., Engineering organization-based multiagent systems, in *SELMAS 2005*, edited by A. G. et al., LNCS 3914, pp. 109–125, 2006.

DeLoach, S. A., Wood, M. F., and Sparkman, C. H., Multiagent systems engineering, *The International Journal of Software Engineering and Knowledge Engineering*, *11*, 231–258, 2001.

Ferber, J. and Gutknecht, O., A meta-model for the analysis and design of organizations in multi-agent systems, in *Third International Conference on Multi-Agent Systems*, pp. 128–135, IEEE Computer Press, 1998.

Ferber, J., Gutknecht, O., and Fabien, M., From agents to organizations: An organizational view of multi-agent systems, in *AOSE 2003*, edited by P. G. et al., LNCS 2935, pp. 214–230, 2004.

Ferraiolo, D. and Kuhn, D., Role-based access control, in *15th National Computer Security Conference*, pp. 554–563, 1992.

Fowler, M., Dealing with roles, in *Proceedings of PLoP '97, Technical Report WUCS-97-34*, Washington University Dept. of Computer Science, 1997.

Jamison, W. C. and Lea, D., Truce: Agent coordination through concurrent interpretation of role-based protocols, in *Proceedings of Coordination 99*, 1999.

Moraitis, P. and Spanoudakis, N. I., Combining gaia and jade for multi-agent systems development, in *Proceedings of the 17th European Meeting on Cybernetics and Systems Research (EMCSR 2004)*, 2004.

Neruda, R. and Beuster, G., Emerging hybrid computational models, in *Proc. of the ICIC 2006*, LNCS 4113, pp. 379–389, 2006.

Neruda, R. and Beuster, G., Toward dynamic generation of computational agents by means of logical descriptions, *International Transactions on Systems Science and Applications*, pp. 139–144, 2008.

Ubayashi, N. and Tamai, T., Roleep: Role based evolutionary programming for cooperative mobile agent applications, in *Proccedings of International Symposium on Principles of Software Evolution*, 2000.

Weiss, G., ed., *Multiagent Systems*, MIT Press, 1999.

Wooldridge, M., Jennings, N. R., and Kinny, D., The gaia methodology for agent-oriented analysis and design, *Journal of Autonomous Agents and Multi-Agent Systems*, *3*, 285–312, 2000.

Xu, H., Zhang, X., and Patel, R. J., Developing role-based open multi-agent software systems, *International Journal of Computational Intelligence Theory and Practice*, *2*, 2007.

Zhang, Z. and Zhang, C., *Agent-Based Hybrid Intelligent Systems*, Springer Verlag, 2004.