Importance and Properties of Roles in MAS Organization: A review of methodologies and systems

Ioannis Partsakoulakis and George Vouros
Department of Information and Communication Systems Engineering
University of the Aegean
83200 Samos, Hellas
{jpar,georgev}@aegean.gr

June 1, 2002

Abstract

Roles have been used both as an intuitive concept in order to analyze multi-agent systems and as a formal structure in order to implement coherent and robust teams. The extensive use of roles in implemented systems evidences their importance in multi-agent systems design and implementation. In this paper we emphasize the importance of roles for multi-agent systems to act in complex domains, identify some of their properties and we review work done concerning specification and exploitation of roles in agent-oriented system engineering methodologies and in multi-agent systems that are deployed in dynamic and unpredictable domains.

1 Introduction

Multi-agent systems (MAS) comprise agents that can be cooperative or self-interested. Cooperative agents coordinate among themselves towards achieving a common goal, where self-interested agents have individual interacting goals. We consider that each agent in a multi-agent system is characterized by a degree of autonomy. Autonomy provides agents with the ability to decide on their own behavior. Given the autonomous nature of each agent, agents need to coordinate among themselves, or else, the group quickly changes to a number of individuals with chaotic behavior. An effective way to achieve coordination is via imposing a specific group organization. An organization comprises roles and their interrelations. A role clusters types of behavior into a meaningful unit that contributes to the group's overall goal. Role interrelations (e.g. hierarchical or class-membership relations) provide communication paths among agents and role interdependencies (e.g. temporal or resource dependency relations) can be exploited for effective agents' coordination and communication. Agents may become members of such an organization, if they succeed to undertake one or more roles that contribute towards the collective objectives.

Collaboration is a special type of coordinated activity, one in which participants work jointly with each other, together performing a task or carrying out the activities needed to achieve a shared goal [8]. Coordination policies and the way they are implemented in a MAS, impact agents' collaborative activity and communication. Generic models that have been devised such as the SharedPlans model [8, 7], the Joint Intentions [2, 11] and Joint Responsibility models [10], provide the principles that underpin social activity and reasoning, and describe the necessary constructs for defining cooperative and individual behavior of agents in social contexts.

Implemented systems [18, 10, 22, 9], aim to make explicit the cooperation model upon which agents' behavior is based. The objective is to provide flexibility towards solving problems related to [10] "*how individual agents should behave when carrying out their local activities in the overall context of action*", "*how the joint action may come unstuck*", "*how problems with the joint action can be repaired*", "*how individuals should act towards their fellow team members when problems arise*" and "*how agents should reorganize their local activity in response to problems with the joint action*". To address these concerns, implemented systems, driven by the high-level cooperation models that they implement, employ constructs and methods such as the intentional context, common recipes [10], fixed organizations with discrete roles interchanged among agents [17], and dynamic assignment of agents to pre-specified roles in conjunction with plan monitoring and repair [22]. The aim is to provide the means for systems to track the mental state of individual agents participating in the cooperative activity in a coherent and integrated way.

The objective of this paper is to show the importance of roles for flexible problem solving by MAS that are deployed in dynamic and unpredictable environments with high degree of interaction and distributivity. The paper aims to identify important properties of roles, examine how roles are conceived in methodologies for implementing MAS and how they are exploited in implemented systems.

The paper is structured as follows: The second section describes the importance of roles in order multi-agent for systems to deal with the complexities of a task-environment and identifies important properties of roles. Section three reports on the way roles are conceived in the context of agent-oriented software engineering methodologies, as well as in multi-agent system architectures and frameworks. The paper concludes with a discussion on the advantages and disadvantages of the several approaches for role specification and exploitation.

2 Importance and properties of roles

In the last few years, the range of applicability of agent technology increased substantially. Recent multi-agent systems (MAS) are deploying in more and more complex environments, including the RoboCup-Rescue [5] and RoboSoccer [14, 15] domains, multi-robot space explorations, battlefield simulations [22, 21] and information integration [20]. The complexity of the environment in which a multi-agent system is deployed and the complexity of the tasks that is expected to perform can be assessed by studying the task-environment in three dimensions [4]: (a) the degree of interaction (b) the dynamics of the environment, and (c) the degree of distributivity.

The *degree of interaction* specifies at what extend the actions and the decisions of agents impact other agents' goals and plans. Interaction results from the necessity to settle issues about (i) limited shared resources, (ii) agents' task interdependencies, (iii) goals and tasks that require collective effort and are shared by a group of agents.

Dynamics specify at what extend the environment changes due to agents' actions or due to environmental changes. High degree and unpredictability of environmental changes, limited agents prediction capabilities and restricted monitoring capabilities complicates copying with the dynamics of environmental changes.

Distributivity specifies at what extend the resources available to the agents are distributed among them. Resources include knowledge about tasks that must be performed, information about the environment and other agents, and other task-specific resources that are inherently distributed to subsets of agents. Agents' capabilities and task knowledge impact distributivity as well. In general, high distributivity complicates attaining consistent and coherent views of the performed tasks, agents' behavior, agents' states and the status of the environment.

2.1 The Importance of Roles

As already defined in the introduction, a role clusters types of behavior into a meaningful unit that contributes to an overall goal. Therefore, roles are considered in the context of achieving specific goals. Roles with their interdependencies and relations in the context of a specific goal specify a method (recipe) for achieving this goal under certain conditions.

The importance of roles in dynamic and unpredictable environments with high degree of interactions and distributivity is as follows:

High degree of interaction

As already stated, interaction results from the necessity concerning settling issues about (i) limited shared resources, (ii) agents' task interdependencies, (iii) goals and tasks that are shared by a group of agents and require collective effort.

The degree of interaction in a team of agents can be reduced by imposing a specific organizational structure on team members, specifying the resources that each agent can use, information that should be communicated between agents, and goals that should be achieved by each agent towards organization's collective objectives. Roles provide the appropriate level of abstraction for the specification of the abovementioned aspects concerning organizations.

Following this approach, coordination can be simplified and the degree of interaction can be reduced, or be controlled effectively.

Environment dynamics

Agents need to deliberate effectively by evaluating their options and opportunities towards achieving their shared goals without considering low-level details of their plans. To plan and act robustly in a dynamically and unpredictably changing environment, agents must reason about their intended behavior in an abstract way. Roles provide such an abstraction, since they aggregate intentions that agents must adopt for the successful execution of tasks, specify the conditions that must hold for roles to apply in a context of action and capture dependencies among the intended behaviors in a shared context.

In dynamic environments, agents must exploit role specifications to deliberatively form organizational structures, revise the existing organizational structure and deliberatively assign roles to agents.

For example, an agent that has been assigned a specific role in the context of a method for achieving a task may recognize its inability to proceed due to an unpredictable lack of resources. In order agents to repair their group activity, they must have an explicit specification of the role each agent plays in the group, as well as the conditions under which a role can be assigned to an agent. Furthermore, agents need a mechanism to dynamically assign roles to them by exploiting role specifications and the overall context of action. Activity repair may result in revising the method employed for achieving the task and/or further agents' group reorganization.

Concluding the above, to deal with environment and task dynamics in a robust way, it requires groups of agents to dynamically assign agents to roles and deliberate on the organization of the group. The reorganization of the group may result not only to new assignments of agents to roles, but to changes in the set of roles, changes in the set of agents forming the group, as well as changes to the number of agents that fill each specific role.

Distributivity

In inherently distributed tasks and environments, agents need to deliberate on who-shall-perform-what so as to manage the distributed nature of the environment and task. Agents need to form groups with shared objectives, interact among themselves so as to have a coherent and integrated view of the whole task-environment and an integrated view of the mental states of their collaborators.

Roles in specific contexts of action provide abstract specifications of distributed behavioral patterns. When these roles are performed in a coordinated fashion, they can accomplish a specific objective. Explicit specification of roles, of their relations and interdependencies provide a shared context for a group of agents to track the mental states of their collaborators and the tasks performed.

However, for agents to plan effectively and manage distributivity in dynamic and unpredictable environments, they should manage roles effectively, decide on the number and types of roles that should be involved, on the number of agents that should fill a role, as well as on the number of roles that a (group of) agent(s) should play.

Summarizing the above, explicit specification of roles and of their interdependencies enable agents (a) to tame the amount of interaction required for effective group behavior by imposing an organization structure, (b) to deal with the dynamics of the task-environment by organizing the group deliberatively and revising the existing organization structure according to their needs, and finally, (c) manage the distributivity of the task-environment by deciding on the assignment of the roles to agents.

2.2 Role Properties

Important role properties that enable agents to deal with the complexity of the task-environment, are the following:

1. Explicit specification of roles, their dependencies and conditions in the context of specific methods for achieving goals.

2. Dynamic assignment of roles to agents in a deliberative way by considering conditions of roles, agents' capabilities and the overall context of action. As already discussed this is important for group reorganization and for managing distributivity.

3. Dynamics of roles. This is important for agents to decide which roles should be employed in an organizational structure for achieving specific goals. Devising dynamic organizational structures deliberatively requires agents to decide on the roles that should be employed. The aim is to reduce interactions among agents and manage distributivity.

4. Cardinality of roles. This specifies the number of agents that should play a role, as well as the number of roles that a single agent should play. In general, having no cardinality restrictions allows agents to flexibly devise more effective organizational structures according to the needs of the tasks and environments, and take full advantage of the capabilities and resources of agents.

5. Lifespan of roles characterizes the dynamics of roles. For instance, agents may dynamically build an organizational structure whose roles do not change during group action. However, revising the assignment of roles to agents or deactivating existing roles, it is important for agents to deal with the dynamics of the environment and handle distributivity effectively.

According to the above, properties of roles for collaborative action in highly dynamic and unpredictable environments with a high degree of interactions and distributivity are the following:

Property	Range of values
Explicit Specification	(Specification of roles)
Static vs. Dynamic	Static to Dynamic
Assignment	Static to Dynamic
Cardinality (roles-to-agents)	One-to-One, One-to-Many, Many-to-Many
Lifespan	Transient to Long-Lived

3 Methodologies and Systems

Roles have been used in MAS extensively, but there is not a methodology or an implemented architecture/framework that satisfies all the requirements described above for building MAS that act in dynamic and unpredictable environments.

In the following we consider methodologies and systems for the specification and exploitation of roles in well-coordinated MAS.

3.1 Roles in AOSE Methodologies

Agent oriented software engineering is central to realizing agent technology as a new software engineering paradigm [24]. Many researchers dealing with the invention of new software development techniques suitable for MAS have conceived multi-agent systems as organizations of agents that interact to achieve common goals. Roles have been used in such methodologies as an intuitive and natural concept for defining agent organizations. For the AAII, Gaia and MaSE methodologies that we discuss subsequently, a role is an abstract entity that is used during system analysis, but does not have any direct realization within the implemented MAS.

3.1.1 MaSE Methodology

MaSE is a methodology for analyzing and designing heterogeneous multi-agent systems. The methodology is supported by the agentTool [3]. The ultimate goal of MaSE and agentTool is the automatic generation of code that is correct with respect to the original system specification.

The MaSE methodology comprises seven steps:

1. Capturing goals: The designer during this step takes the initial system specification and transforms it into a structured set of system goals, depicted in a goal hierarchy diagram.

2. Applying use cases: During this step, use cases drawn from the system requirements are structured as sequence of events and are used in translating goals into roles. This step results in an initial set of roles. A role in MaSE is an abstract description of an entity's expected function.

3. Refining roles: In this step the designer ensures that all the necessary roles have been identified and develops the tasks that define role behavior and communication patterns. Roles are captured in a Role Model specifying roles and their interrelations. Specification of roles comprises their identity and tasks that roles must perform for accomplishing their goals.

4. Creating agent classes: During this step, the designer specifies the agent classes that are identified from roles specifications. Agent classes are defined in terms of roles they play and conversations in which they must participate. These classes are documented in an Agent Class Diagram.

5. Constructing conversations: In this step, coordination protocols between pairs of agents are been constructed.

6. Assembling agent classes: This is the step where the internals of agent classes are created.

7. System deployment: This is the final step where the designer defines the configuration of the actual system to be implemented.

The designer is expected to move back and forth between these steps to ensure models and diagrams that are produced are complete and consistent among themselves.

Roles in MaSE are used in the analysis phase and are substituted by the agent classes in the design phase. Therefore, roles are not realized in the final system implementation.

A role in MaSE is an abstract specification of an entity's expected function that comprises the tasks that should be performed for the role to accomplish its goals. The specification and interrelations of roles are defined by the designer during the design phase and do not change. Since each agent class is defined by the set of roles it plays, roles' lifespan equals to the lifespan of the agents in the system. Furthermore, the assignment of agent classes to roles is done during the design phase. Generally, as mentioned in [3], although there is a one-to-one mapping between roles and agent classes, the designer may combine multiple roles in a single agent class or map a single role to multiple agent classes. Agents, as instances of agent classes, do not deliberate on role assignments and therefore, are not able to revise the devised organization.

Role properties in MaSE are summarized in the following table:

Property	Range of values
Specification	A role is not realized in the final system implementation. It is an abstract description of an entity's expected function that comprises tasks that should be performed
Assignment	Static. No deliberative assignment
Static vs. Dynamic	Static. Agents cannot decide which roles should be employed
Cardinality	Many-to-Many. Many agents can play a role. Each agent plays an instance of that role
Lifespan	Long-Lived

Concluding the above, systems developed with the MaSE methodology, although they can control interaction by means of agents' coordination protocols in a fixed organization structure, they cannot cope with environment dynamics since roles are not defined explicitly in the final system realization. For instance, roles do not provide the abstractions needed for agents robust planning and execution, and do not enable agents to deliberate on roles assignment. Therefore, agents can not revise the given organizational structure. This also affects distributivity, since agents cannot form groups at need in a dynamic fashion.

3.1.2 Gaia Methodology

Gaia is intended to allow an analyst to go systematically from a statement of requirements to a design that is sufficiently detailed that it can be implemented directly [25]. Gaia, as it is stated in [24], encourages developers to think of building agent-based systems as a process of organizational design. An organization is considered to be a collection of roles that stand to certain relationships to each other.

A role in Gaia is defined by four attributes: *responsibilities*, *permissions*, *activities*, and *protocols*. Responsibilities determine the functionality of the role and are divided into *liveness* properties and *safety* properties. Liveness properties describe those states of affairs that must be brought about by the agent that has been assigned to the role. Safety properties describe those states of affairs that the agent must maintain across all states of execution of the role. In order to realize responsibilities, a role has a set of permissions that identify the resources that are available to that role. Activities are computations associated with a role and may be carried out by the agent without interacting with other agents. Finally, protocols define role interactions.

The analysis stage of Gaia is based on the following steps:

1. Identification of the roles in the system. This gives a prototypical (informal and unelaborated) role model, defined as a set of role schemata. Each schema comprises protocols, activities, permissions and responsibilities.

2. Protocols of interactions among roles are identified and documented. This results in an interaction model, which captures the recurring patterns of inter-role interaction for tasks performance.

3. Full elaboration of roles model, using the protocol model.

Roles in Gaia are specified using the role schemata, but they are not explicitly defined in the actual system implementation. During the design phase role specifications guide the definition of agent types in the system. Roles are static and long-lived, because they define a fixed organizational structure and are statically associated with specific agents types. However, one or more instances of an agent type (which plays an associated role) can be created according to instance qualifiers introduced in the agent model. Instance qualifiers define the cardinality of roles, i.e., the number of agents of the same class that can be assigned in a specific role. However, agents cannot deliberate on the organizational structure and on the number of agents that should play a specific role.

The following table describes the properties of roles, as these are considered in Gaia.

Property	Range of values
Specification	A role is viewed as an abstract description of an entity's expected function and is characterized by permissions, protocols, activities and responsibilities
Assignment	Static. Agents cannot deliberate on roles assignment
Static vs. Dynamic	Static. Agents cannot decide which roles should be employed
Cardinality	Many-to-Many. Many agents can play the same role. Each agent plays an instance of that role
Lifespan	Long-Lived

Concluding the above, systems developed with the Gaia methodology have a fixed organization structure managing interaction and distributivity in a static fashion. Such systems cannot cope with environment dynamics since roles are not defined explicitly in the final system realization. Roles, although elaborated and defined in a detailed way during design, do not provide the abstractions needed for agents' robust planning and execution, and do not enable agents to deliberate on roles assignment.

3.1.3 AAII Methodology

The AAII methodology [12] operates at two levels of abstraction, the external and the internal viewpoint. From the external viewpoint the system is decomposed into agents, modelled as complex objects characterized by their purpose, their responsibilities, the services they perform, the information they require and maintain, and their external interactions. From the internal viewpoint, beliefs, goals and plans must be specified for each agent. Therefore, agents in systems built with AAII are considered to follow the Belief-Desire-Intention (BDI) paradigm.

The details of the external viewpoint are captured in the agent and interaction models. Their elaboration and refinement is done in four major steps.

1. Identification of the roles that apply and elaboration of an initial agent class hierarchy. Roles can be organizational or functional. I.e., they can be related to the application, or they can be required by the system implementation, respectively.

2. Identification of roles' responsibilities and of those services needed for fulfilling responsibilities. Services are activities that are not decomposed further and may include interaction with the external environment or other agents. Agent classes are further decomposed to the service level.

3. Identification of interactions associated with services. This includes the identification of performatives (speech acts), information content, events and conditions to be noticed, actions to be performed, and other information requirements for interactions. This step includes determination of the control relationships between agents. At this point the internal modeling of each agent class can be performed.

4. Final refinement of the agent and control hierarchies and introduction of the agent instances.

The methodology for internal modeling begins from the services provided by an agent and the associated events and interactions. These define the purpose and the top-level goals of an agent. Internal modeling can be expressed at two steps.

1. Analysis of the means for achieving the goals. This includes, contexts, conditions, subgoals, actions, and handling of failures. It results in a plan for achieving each goal.

2. Consideration of the beliefs that affect the appropriateness of a given plan and the manner in which it is carried out in various contexts. Analysis of the input and output data requirements for each subgoal in a plan.

The final system realizes the agent hierarchy that is built based on roles and role interactions. This hierarchy and the corresponding roles do not change during group performance. In accordance to the previous methodologies, roles are not realized in the final system but drive the implementation of the agent classes.

The properties of roles for AAII are summarized in the following table:

Property	Range of values
Specification	A role is specified by a set of responsibilities, services and interactions among services
Assignment	Static
Static vs. Dynamic	Static. Agents cannot decide which roles should be employed
Cardinality	Many-to-Many. Many agents can play a same role. Each agent plays an instance of that role
Lifespan	Long-Lived

Concluding the above, agents employed in systems developed with the AAII methodology comply with a rigid organizational structure, but in correspondence with the above mentioned methodologies cannot cope with environment dynamics since roles are not defined explicitly in the final system realization. This affects robust planning and execution, effective management of distributivity and interaction.

3.1.4 AALAADIN Model

AALAADIN is not a specific agent methodology, but a meta-model for describing organizations of agents using the core concepts of group, agent and role [6]. The methodology is supported by the MadKit platform. With AALAADIN one can describe multi-agent systems with different forms of organizations such as market-like or hierarchical organizations and therefore it could be useful for designing MAS. An organization in AALAADIN is a framework for activity and interaction through the definition of groups, roles, and their relationships.

A group is defined to be an atomic set of agent aggregation and has a finite set of roles. Groups and roles are specified using the MadKit platform during systems development. A group can be created by any agent that is then automatically takes the special role of group manager. The group manager has the responsibility for handling requests for group admission or role requests.

A role in this model is an abstract representation of an agent function, service or identification within a group. Each agent can take several roles, and each role assigned to an agent is local to a group.

The most interesting feature of AALAADIN is that an agent can create new groups with roles. Therefore, the structure of the system organization can be constructed dynamically. Agents can create roles with transient lifetime in a group structure. AALAADIN does not aim at cooperative behavior. It just provides to the designer the above-mentioned facilities in order to build groups of agents.

Building a MAS with AALAADIN requires substantial effort from the system designer and the properties of roles depend on the sophistication of the designed system.

The following table illustrates the properties of roles in AALAADIN.

Property	Range of values
Specification	A role is viewed as an abstract representation of an agent function, service or identification
Assignment	Dynamic
Static vs. Dynamic	Dynamic
Cardinality	Many-to-One
Lifespan	Trnasient (it can be long-lived)

Concluding the above, the AALAADIN model provides developers with the ability to define systems that can cope with task-environment dynamics in terms of agents groups and roles. The actual properties of roles depend on the final MAS system design sophistication. However, for complex environments it requires substantial effort from the system designer and developer in order MAS to realize the full-range facilities provided by roles. This is mainly due to the fact that there is not a specific methodological and/or development framework for the specification and exploitation of roles during planning and execution of tasks.

3.2 Roles in MAS

Roles have been used in implemented multi-agent systems in order to achieve coherence in teams of cooperative agents in domains where well-coordinated activity is required. Such domains include battlefield simulations [21] and the robocup simulation league [13]. Moreover this section refers to frameworks for implementing cooperative agents following the role-oriented agent-programming paradigm [1, 16].

3.2.1 The Karma-Teamcore framework

The Karma-Teamcore framework focuses on rapidly integrating distributed, heterogeneous agents and tasking them via an abstract team-oriented program [23]. The framework provides wrappers that encapsulate general teamwork reasoning and automatically generate the necessary coordination for robust execution.

In this framework a system developer builds a team-oriented program that consists of a team organizational hierarchy, a team (reactive) plan hierarchy and assignments of agents to roles. Roles are used in the specification of the organizational hierarchy, which does not change during team performance. Roles are assigned to specific tasks in the plan hierarchy and agents can play specific roles depending on their capabilities. Roles in this framework are abstract specifications of a set of activities. Therefore roles and their interrelations are static. They provide a useful level of abstraction for (re)assigning agents to tasks based on monitoring and re-planning mechanisms. Such mechanisms are either generic, exploiting role relationships (AND, OR Role-dependency), or partly domain dependent for inferring role (non) performance. It must be pointed that the (re)assignment of agents to roles is based on the capabilities that each role requires without agents to consider the overall context of action.

Leaf nodes in the organizational hierarchy correspond to single agents. The internal nodes in the organizational hierarchy correspond to groups of agents (group roles) that are defined implicitly by their successor leaf nodes. Tasks that are assigned to a group role is assigned to each member of the group that corresponds to this role. Roles do not change during group action and therefore are long-lived.

The properties of roles in this approach are shown in the following table.

Property	Range of values
Specification	A role is an abstract specification of a set of activities and inherits the requirementsfrom each plan it has been assigned
Assignment	Dynamic. Reactively based on agents' capabilities
Static vs. Dynamic	Static. Agents cannot decide which roles should be employed
Cardinality	Many-to-Many. Many agents are assigned to a group role and each agent may play more than one role
Lifespan	Long-Lived

Concluding the above, the Karme-Teamcore provides a framework for the specification of a fixed organizational structure in terms of roles that agents should play for the achievement of specific goals. Roles provide a valuable abstraction for assigning tasks to agents. Therefore, roles provide the means for dealing with environment dynamics. This is achieved by monitoring and replanning agent abilities based on roles, and via agents reassignment to roles. However, agents do not deliberate on roles assignment (considering the overall context of action) but they are rather reactively assigned to roles based only on their capabilities. Agents cannot revise the given organizational structure. This may affect distributivity in cases that agents cannot be assigned roles from the prespecified organizational structure because of their limited capabilities. In these cases alternative organizations may lead the team to a successful execution of its task. As far as managing interactions is concerned, Karma-Teamcore integrates a decision theoretic communication selectivity mechanism based on communication costs and benefits. However, roles are not exploited for managing interactions.

3.2.2 The RoboCup Simulation Domain

RoboCup simulation [13] is a highly dynamic, real-time environment with many real-world complexities. In this domain there are two teams of agents consisting of eleven members each. The objective of a team is to win the game, i.e. to score more goals than the opponent team. In order to fulfill this objective, the team players must act in a well-coordinated and coherent fashion. Roles have been used in the most well-known team architectures, that of the CMUnited [19] that won the RoboCup world championships RoboCup98 and RoboCup99 and that of FC Portugal [17] the winner of the RoboCup2000 world championship. A role in robotic soccer can be as simple as a position in the field.

In the CMUnited [19] architecture an agent has a set of internal and external behaviors. Internal behaviors update the agent's internal state, while external behaviors reference the world and agents' internal states and select the actions to be executed. Internal and external behaviors are both sets of condition/action pairs. Conditions are logical expressions and actions are behaviors themselves.

A role is a specification of agent's position and inter-position behavior, like passing options. Therefore, a role aggregates a cluster of behaviors in a logical group, which specifies agent's internal and external behaviors. Roles are static and long-lived and are grouped into formations. Agents can change formation at run-time. This results in changing the characteristics of roles at run-time. However, agents within a formation can also inter-change roles in order to save energy [17]. An agent undertakes one role in each formation.

The next table summarizes the role properties in these systems.

Property	Range of values
Specification	A role is a specification of agent's position and inter-position behavior
Assignment	Merely dynamic by changing positions in the field
Static vs. Dynamic	Merely dynamic by deciding which strategy to follow
Cardinality	One-to-One
Lifespan	Transient (strategies may define different agent types)

Concluding the above, environmental changes in Robocup are due to agents' actions and decisions. Agents in a team can reactively follow pre-specified organizational structures, which may change during a play. Such organizations comprise eleven roles, each for a single agent. These roles can be interchanged between agents in a reactive way (depending on the current situation). The main issue concerning RoboCup is that it provides a case study were roles have been successfully employed for coordinating agents to reactively achieve their tasks.

3.2.3 Role Oriented Programming

A multi-agent system with a dynamic organization can change in size and structure, dynamically (re)assign agents to roles, decide on the number of agents that should fill a role and deliberate on the roles that should structure the group. ROPE framework [1] as well as the work reported in [16] aim to this target by the use and exploitation of roles for specifying cooperation processes. Both works give a strong emphasis on the role concept.

In ROPE a role is defined as an entity consisting of a set of required permissions, a set of granted permissions, a directed graph of service invocations and a state not visible to other agents. The service invocations describe the agent behavior and may contain an arbitrary number of alternatives. Roles may have an associated set of sub-roles that inherit granted permissions. An agent in ROPE is defined as a set of provided services.

The aim in [16] is to build a generic framework for implementing collaborative agents. Roles are used to define the intended behavior of an agent within a collaborative task. Tasks are defined within a special formal structure called a multi-role recipe. The recipe constitutes the know-how of each agent and contains roles-specifications. Each role specification comprises capabilities that the agent must have in order to undertake each role, a number of constraints that must be preserved during the execution of the cooperative task, an action list as well as temporal and synchronization constraints among actions, and a number of effects that are achieved when the role has been executed successfully. An agent can undertake one or more roles either in the context of the same activity or in different activity contexts. A role can be undertaken by a group of agents.

These approaches seem to exploit most of the dynamics of roles in complex multi-agent systems. Agents may form organization structures that are dynamic, as roles can be discarded or changed at run-time. The lifespan of roles is transient (roles are used when needed and for as long as required) and the (re)assignment of roles is dynamic. In both approaches an agent can undertake more than one role in different goal contexts. However, in the second approach a single role can be assigned to a group of agents. In this case, a task can be delegated to a group of agents who must contact further planning to decide on the way to perform their task.

The following table summarizes the role properties of both role-oriented programming approaches.

Property	Range of values
Specification	Set of activities for achieving a specific goal, associated with constraints (contextual, capability and mental state) and effects
Assignment	Dynamic, including deliberation in the overall context of action
Static vs. Dynamic	Dynamic. Agents decide which roles should be employed
Cardinality	Many-to-Many. Many agents assigned to a role means either that each agent has a copy of that role or that the agents collectively execute that role
Lifespan	Transient

Role-oriented agent-programming provides the full range of facilities for agents to manage the complexity of highly dynamic and unpredictable environments with a high degree of interaction and distributivity. Agents can deliberatively devise an organizational structure, decide on the assignment of roles to agents and revise their decisions, in parallel to planning and executing their tasks. This provides them with the ability for flexible problem solving. Moreover, it affects distributivity, since agents may form groups at need in a dynamic fashion. Finally, explicit specification and shared knowledge of roles and their interrelationships in a context of action enables agents to manage interactions.

4 Concluding remarks

As we saw earlier, roles have been used both as an intuitive concept in order to analyze multi-agent systems and as a formal structure in order to implement coherent and robust teams. In this paper we presented the role-related work on agent-based systems done in two main streams of research, that of agent-oriented system engineering and that of implemented multi-agent systems that are deployed in complex domains. The following table illustrates the several approaches in using roles and their role-related properties.

Property	MaSE, Gaia, AAII	AALAADIN	Teamcore	Robocup	ROP
Specification	Roles are not realized in the final system. They are specified as abstract behaviors and are used to define abstract agent types or classes	A role is viewed as an abstract representation of an agent function, service or identification, service or identification	A role is an abstract specification of a set of activities and inherits the requirements from each plan it has been assigned	A role is a specification of agent's position and inter-position behavior	Set of activities for achieving a specific goal, associated with constraints (contextual, capability and mental state) and effects
Assignment	Static. No deliberative assignment	Dynamic	Dynamic. Reactively based on agents' capabilities	Merely dynamic by changing positions in the field	Dynamic, including deliberation in the overall context of action
Static vs. Dynamic	Static. Agents cannot decide which roles should be employed	Dynamic	Static. Agents cannot decide which roles should be employed	Merely dynamic by deciding which strategy to follow	Dynamic. Agents decide which roles should be employed
Cardinality	Many-to-Many. Many agents can play a role. Each agent has a copy of that role	Many-to-One	Many-to-Many. Many agents are assigned to a group role and each agent may play more than one role	One-to-One	Many-to-Many. Many agents assigned to a role means either that each agent has a copy of that role or that the agents collectively execute that role
Lifespan	Long-Lived	Transient (it can be long-lived)	Long-Lived	Transient (strategies may define different agent types)	Transient

From this table we can observe that agent-oriented software engineering is in an early stage concerning the analysis and specification of systems that act in dynamic and unpredictable environments. Therefore, a lot of work has to be done in this area. Until now, the most well known methodologies are concerned with the analysis and design of systems with a fixed number of agents, with a static structure and fixed inter-agent relationships and dependencies. In such systems there is no need for introducing role structures dynamically and reasoning about roles. Agent-oriented methodologies concern about the flexibility and robustness of the resulting implemented agent systems in a limited manner, without dealing with those aspects of the problem domains that concern dynamic assignment of tasks to agents, the run-time selection of roles, as well as the reorganization of the overall system. These restrictions result from the fact that these methodologies do not realize roles and their interrelations during the design and implementation of the multi-agent system.

Implementation-specific works are actively being concerned with more complex environments and therefore, with advanced role properties. The extensive use of roles in implemented systems evidences the need for role-oriented thinking and modelling in multi-agent design and implementation. However, there is not an agent framework, an implemented system or architecture that exploits the full range of facilities provided by roles in an integrated fashion.

We are currently working on the analysis and design of a generic role-based model of collaborative activity. The development of the architecture presented in [16] is a major step towards this target. The full-range development of this architecture is ongoing work.

References

[1] M. Becht, T. Gurzki, J. Klarman, and M. Muscoll. ROPE: Role Oriented Programming Environment for Multiagent Systems. In *Proceedings of the Fourth IECIS International Conference on Cooperative Information Systems*, 1999.

[2] Philip R. Cohen and Hector J. Levesque. Teamwork. *Nous*, 25(4):487–512, 1991.

[3] Scott A. DeLoach and Mark Wood. Developing Multiagent Systems with agentTool. In C. Castelfranchi and Y. Lesperance, editors, *Intelligent Agents VII*, LNAI 1986, pages 46–60. 2001.

[4] Edmund H. Durfee. Scaling Up Agent Coordination Strategies. *IEEE Computer*, pages 39–46, July 2001.

[5] H. Kitano et al. Robocup-Rescue: Search and Rescue for Large Scale Disasters as a Domain for Multi-Agent Research. In *Proceedings of the IEEE Conference on Man, Systems, and Cybernetics (SMC-99)*. 1999.

[6] Jacques Ferber and Olivier Gutknecht. A meta-model for the analysis and design of organizations in multi-agent systems. In *Proceedings of the Third International Conference on Multi-Agent Systems*, 1998.

[7] Barbara Grosz and Sarit Kraus. The evolution of SharedPlans. In Anand Rao and Michael Wooldridge, editors, *Foundations and Theories of Rational Agencies*. Kluwer Academic Press, 1999.

[8] Barbara J. Grosz and Sarit Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, October 1996.

[9] Merav Hadad and Sarit Kraus. SharedPlans in Electronic Commerce. In M. Klusch, editor, *Intelligent Information Agents*, chapter 9, pages 204–231. Springer, 1999.

[10] Nicholas Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence*, 75, 1995.

[11] D. Kinny, M. Ljungberg, A. Rao, E. Sonenberg, G. Tidhard, and E. Werner. Planned Team Activity. In C. Castelfranchi and E. Werner, editors, *Artificial Social Systems*, LNAI 830. 1992.

[12] David Kinny, Michael Georgeff, and Anand Rao. A Methodology and Modelling Technique for Systems of BDI Agents. In *Agents*

Braking Away, Seventh European Workshop on Medelling Autonomous Agents in a Multi-Agent World, MAAMAW'96, LNAI 1038. 1996.

[13] H. Kitano, M. Tambe, M. Veloso, I. Noda, E. Osawa, and M. Asada. The robocup synthetic agents' challenge. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1997.

[14] Itsuki Noda. Soccer server: A simulation of robocup. In *Proceedings of AI symposium '95 Japanese Society for Artificial Intelligence*, pages 29–34, 1995.

[15] Itsuki Noda, Hitoshi Matsubara, Kazuo Hiraki, and Ian Frank. Soccer server: A tool for research on multiagent systems. *Applied Artificial Intelligence*, 12:233–250, 1998.

[16] Ioannis Partsakoulakis and George Vouros. Roles in Collaborative Activity. In I. Vlahavas and C. Spyropoulos, editors, *Methods and Applications of Artificial Intelligence, Second Hellenic Conference on AI*, LNAI 2308, pages 449–460. 2002.

[17] Luís Paulo Reis, Nuno Lau, and Eugénio Costa Oliveira. Situation Based Strategic Positioning for Coordinating a Team of Homogeneous Agents. In M. Hannebauer, J. Wendler, and E. Pagello, editors, *Balancing Reactivity and Social Deliberation in Multi-Agent Systems*, LNAI 2103, pages 175–197. 2001.

[18] Charles Rich, Candace Sidner, and Neal Lesh. COLLAGEN: Applying Collaborative Discource Theory to Human-Computer Interaction. *AI Magazine*, 22(4):15–25, 2001.

[19] Peter Stone and Manuela Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 110:241–273, 1999.

[20] K. Sycara, M.Paolucci, M van Velsen, and J. Giampapa. The RETSINA MAS Infrastructure. Technical Report CMU-RI-TR-01-05, CMU Technical Report, 2001.

[21] M. Tambe, K. Schwamb, and P. S. Rosenbloom. Constraints and design choices in building intelligent pilots for simulated aircraft pilots for simulated aircraft: Extended Abstract. In *AAAI Spring Symposium on Lessons Learned from Implemented Software Architectures for Physical Agents*, 1995.

[22] Milind Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.

[23] Milind Tambe, David V. Pynadath, and Nicolas Chauvat. Building Dynamic Agent Organizations in Cyberspace. *IEEE Internet Computing*, pages 65–73, March-April 2000.

[24] Michael Wooldridge and Paolo Ciancarini. Agent-Oriented Software Engineering: The State of the Art. In P. Ciancarini and M. Wooldridge, editors, *Agent-Oriented Software Engineering*, LNAI 1957. 2001.

[25] Michael Wooldridge, Nicholas R. Jennings, and David Kinny. The Gaia Methodology for Agent-Oriented Analysis and Design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.