Roles in Collaborative Activity

Ioannis Partsakoulakis and George Vouros

Department of Information and Communication Systems
School of Sciences University of the Aegean 83200, Karlovassi, Samos
{jpar,georgev}@aegean.gr

Abstract. Cooperative activity involves collaboration and communication. Through the stages of collaboration, agents may play different roles either for performing domain tasks, or for forming decisions concerning the collaborative activity itself. Collaboration and communication can be enhanced if dependencies between agents' intentions are captured. Role-specification is expected to be a vital factor towards this goal. This is evidenced by roles' importance in many implemented systems. Agents' coordination, plan monitoring and re-planning in these systems rely on contextual information and agents' roles. However, there is not an implemented generic agent architecture that realizes the importance of roles for flexible cooperative activity. This paper shows how the ICAGENT development framework has been evolved to support cooperative activity through representing and reasoning about multi-role recipes.

1 Introduction

In the last few years, due to the increased degree of complexity in domains where the role of intelligent systems is foreseen and the need to employ systems in complex, dynamic and unpredictable environments, there is a great interest in building multi-agent systems where (homogeneous or heterogeneous) agents are collaborating and communicating towards achieving a shared objective. Examples of multi-agent systems with advanced cooperative abilities can be met in real-time, non-deterministic and dynamic environments such as in the RoboCup-Rescue [3] and RoboSoccer [9, 10] domains, as well as in multi-robot space explorations, battlefield simulations [15, 14] and information integration [13]. In these cases, due to agents' actions interferences and dependencies, agents must be able to coordinate their actions and communicate effectively in all stages of the cooperative activity.

Generic models for collaborative activity such as the SharedPlans model, the Joint Intentions and Joint Responsibility models [1, 4, 6, 7] provide the principles that underpin social activity and reasoning, and describe the necessary constructs for defining cooperative and individual behaviour of agents in social contexts. Intentions play a major role in these models and drive agents to (a) commit to bring about a particular state of affairs, (b) organize and perform appropriate actions within the overall context of action in a coherent and consistent manner and (c) contact means-end reasoning for working the low-level details of their actions.

Implemented systems [12, 6, 15, 5], aim to make explicit the cooperation model upon which agents' behaviour is based. The objective is to provide flexibility towards solving problems related to [6] "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 re-organize their local activity in response to problems with the joint action". Major issues of concern are the following: (a) Coordinating agents' activity towards coherent group action, (b) reducing the amount of communication messages exchanged between agents, and (c) communicating the necessary amount of information at the appropriate time point, so that effective coordination to be achieved.

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 [6], fixed organizations with discrete roles interchanged among agents [11], and dynamic assignment of agents to prespecified roles in conjunction with plan monitoring and repair [15]. 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. Although the importance for the employment of roles and contextual information is well evidenced in implemented systems, there is not an implemented generic agent architecture that provides the full range of facilities for collaboration and communication provided by common recipes and roles.

The objective of this paper is to report on the evolution of the ICAGENT agent development framework, that employs the SharedPlans model, to support cooperative activity through representing and reasoning about multi-role recipes.

The paper is structured as follows: Section 2 motivates our work by presenting previous approaches related to the employment of roles in cooperative activity. Section 3 briefly presents the ICAGENT agent development framework and describes its evolution towards representing and exploiting multi-role recipes for cooperative activity. Finally, section 4 concludes the paper with remarks and future work.

2 Motivation and previous work

Collaborative activity comprises the phases of recognition, in which an agent identifies the potential for collaboration, team formation, in which the agent solicits assistance, plan formation, in which the newly formed team attempts to construct an agreed shared plan, and finally execution, in which members of the team try to achieve the objectives they have committed [16].

It is during team formation that a set of agents shares an intention to achieve an action. Each action is realized by one or more alternative recipes. Recipes comprise conditions for being selected, applicability conditions and, as far as complex actions are concerned - actions that need further planning and refinement - the recipe specifies a sequence of sub-actions that the agent must perform

for completing the plan. A *role* is a specification of a subset of actions that an individual or a team undertakes in the context of a recipe.

For instance, teaching a course, which with no doubt is a complex action, requires careful planning. This means that agents shall form intentions towards their common goal and choose a recipe that achieves courses' objectives in the required context. The context comprises the curriculum in which the course is being taught, the time available for the course, restrictions on the way it shall be examined, the specific programme in which it is being taught. The selected recipe may involve two roles: One for the lecturer and one for the teaching assistant. Each agent has its own responsibilities during cooperative activity: Each one must perform its local activities in the overall context of action, must inform the other in case he/she is not able to perform its role in the context of the joint activity, shall try to provide the necessary resources for the joint task to be performed successfully, and shall try to re-organize the overall activity in case there is any problem with the joint action.

Distinct, well-defined and clear roles, which are motivated in the context of the overall activity, provide agents with information about activities' interdependencies, activities' coordination and communication requirements. For instance, in case the lecturer in our teaching scenario fails to achieve its weakly task, then it must inform the teaching assistant about this failure, since the weakly task of the latter depends on the task of the former. This is evidenced by interdependencies [1] between actions in roles. The assistant, exhibiting helpful behaviour must either perform the task, or in case this is not possible, the team must re-plan its overall activity for the next weeks. However, the teaching assistant must not perform its individual activity, as it would do in case the lecturer had performed its task successfully.

It must be noticed that in case a recipe involves two or more roles, it does not mean that it is necessarily a multi-agent recipe: In case there is not a teaching assistant with the necessary abilities, and if the lecturer's commitments allow, he/she may also be assigned the role of the teaching assistant. In this case the recipe would be performed, as it would be a single-agent recipe.

On the other hand, in case more agents could be employed in the respective roles, for instance, experts in specific areas could give lectures, or assistants - each with a given specialty - provide teaching assistance, then each role may have been filled with a team of agents. In this case, sub teams must cooperate and contact means-end analysis towards performing a single role. This leads to a dynamic organization, in the sense that cooperating agents, or sub-teams of agents, are assigned roles depending on needs and availability, resulting in a hierarchy of roles that is formed in parallel to the joint plan.

According to the above, it is conjectured that roles' specifications in the context of recipes must provide great flexibility for the agents to build teams and cooperate towards achieving their common objectives. Specifically, roles must

[1] Such an interdepedency can be a common parameter, or a temporal relation between actions.

a. Specify the actions that an individual or a team must undertake in the context of a recipe,
b. Facilitate agents to decide whether a recipe shall be utilised as a single or as a multi-agent recipe,
c. Allow agents to decide on the best way for filling roles and performing actions in an arbitrary level of detail. This may result in building dynamic organizations in the sense specified above,
d. Facilitate agents to coordinate their activities during planning and execution, by capturing actions interdependencies,
e. Provide support for effective agents communication, reducing the amount of communication messages and communicating the necessary information. This can be achieved by inferring roles interdependencies from role specifications.

As it is already pointed in section 1, the need for roles and contextual information is well evidenced in implemented systems. However, there is not an implemented agent architecture that provides the full range of facilities for collaboration and communication provided by commonly agreed recipes and roles. The only known system that allows agent developers to specify roles in conjunction with plans is STEAM [15]. The developer has to specify three key aspects of a team of collaborating agents: A team organization hierarchy, a team reactive plan hierarchy and assignments of agents to execute plans. The latter is done by assigning the roles in the organization hierarchy to plans, and then assigning agents to roles by exploiting only agents' capabilities. Agents do not exploit contextual information for this assignment. This is justified by the use of reactive plans: Agents do not deliberate whether they should be committed to an action by reconciling their intentions and desires. Consequently, whether an operator is a team operator or an individual operator is dynamically determined only by agents' capabilities. Developers specify domain-dependent coordination constraints for agents assigned to roles, while domain independent ones are inferred from roles specifications. Role specifications are used mainly for plan performance monitoring and re-planning.

In [5] recipes are specified to be either single or multi-agent. The number of agents in multi-agent recipes is fixed. In this case, variables in recipes represent certain agents. In GRATE* [6], the organizer of the cooperative activity agrees on a common recipe with other team members, and decides which part of the recipe can undertake and which part is going to delegate to other agents. Each agent adopts one or more recipe actions. In GRATE* there are no roles defined in conjunction with recipes. Roles are dynamically identified and assigned to agents by the organizer who exploits the temporal relations between actions. Furthermore, agents are not able to plan in an arbitrary level of detail (planning reaches only the second level). This prohibits agents from planning and building dynamic organization hierarchies at an arbitrary level of detail.

This paper evolves the ICAGENT framework for developing cooperative agents, by providing an enhanced version of recipes that contain role specifications. Roles dynamically define organizational relationships among agents. Our aim

is to provide agents with the necessary flexibility for solving complex problems in dynamic and unpredictable environments in cooperation with other agents, through the definition of multi-role recipes. Agents, depending on context and their mental state may deliberate about role assignment, or be reactively assigned to roles. Therefore, the role, and consequently the task assignment, is done in a flexible way.

3 The Multi-Agent Tileworld domain

The Multi-Agent Tileworld (MAT) [2] is an abstract, dynamic, simulated environment, with embedded agents, developed to support controlled experimentation with agents in dynamic environments.

As it is shown in figure 1, MAT is a chessboard like environment with empty squares, obstacles, holes and agents. Each agent is a unit square with the ability to move in all directions, except diagonally, by one square per move. Holes and obstacles are also unit squares that appear and disappear randomly. Obstacles and tiles have varying weights. Each agent is able to carry tiles whose weight is less than a maximum weight. The expected time for a tile to disappear (TTL) is known and the goal is to fill as many holes as possible in the minimum time.

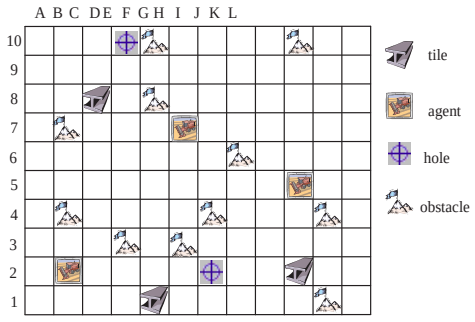

Fig. 1. The MA-Tileworld domain

To show the importance of roles in the cooperative activity, let us consider the following scenario: Assume that agent in 7F desires to fill the hole in 10D. The agent is not able to load tile 8C by himself and the other tiles are too far: The cost associated with these tiles is two high compared to the cost associated with 8C. In this case the agent should check the potential for collaboration. Agent in 7F should ask for the help of other known agents in the MAT in order to load the desired tile. Assume that agent in 2B is able to load 8C. Having agreed on the principle for joint action, the two agents (7F and 2B) must find a common recipe towards the desired state. To collaborate effectively, it is not enough for the agents to commit to a common recipe. Agents must also commit

to specific roles in the context of the common recipe. For instance, the action of loading the tile 8C shall be done by the agent with the corresponding capability, while the action of putting the tile to the corresponding hole shall be done by the originator agent 7F. However, both agents have to move to square 8C. In case any of the agents fails to perform its role successfully, the other must exhibit helpful behaviour, making the best for the completion of their shared activity.

Having agreed on a common recipe and being committed to specific roles in the context of this recipe, agents have already decided that the recipe shall be utilised as a multi-agent recipe.

Proceeding deliberatively, agents decide on the best way for filling roles: Each agent reconciles roles' conditions and restrictions imposed by their common recipe, with constraints holding in its individual context of action and with further desires and commitments it may hold. In case more than one agents fill a recipe role (for instance many agents may help loading the tile), this drives the system to contact further planning, making the roles of these agents discrete. This results in building dynamic organizations in the sense specified above. Agents may also proceed reactively. In this case they do not reconcile the intentions to perform some roles with other intentions and desires they may hold, but they just check the capabilities each role requires.

Having committed to specific roles, agents coordinate their activities by means of actions interdependencies. For instance, agents shall meet in a specific square in the MAT. In case agent 2B arrives first at the meeting point, gets the tile and waits for the agent 7F to arrive there. On the contrary, if agent 7F arrives first, it waits for the other to get there and pick the tile. The context of roles' performance is also crucial for agents' cooperation: In case agents' roles are in the context of a recipe for filling a hole, agent 2B shall wait for the agent 7F to fill the hole, until it knows that their shared activity has been performed successfully. However, in case agents have committed to roles in the context of a recipe for loading a tile, then their joint activity is completed when the agent 7F has the tile been loaded.

In case any of the agents fails to perform its role successfully, it must communicate to the other its failure, as well as any information needed for the other to proceed. For instance, the failure of any agent to perform its role may lead the other to seek for alternative partner(s) or alternative recipes. Furthermore, roles help agents interpret each others' actions: the agent in 7F can interpret the motion of the agent in 2B towards their meeting point.

4 ICAGENT framework and multi-role recipe specification

As already pointed, this paper evolves the ICAGENT generic framework towards the representation and exploitation of multi-role recipes for agents' coordination and communication. ICAGENT allows for the development of agents that reason about their plans and balance between deliberation and reaction. Key issues towards this aim are the following:

– Equip agents with advanced plan management tasks, so that agents are able to balance between reaction and deliberation.
– Provide a clear distinction between deliberation and reaction in terms of agents' reasoning tasks and management of agents' mental state. Agents may contact "careful" planning (deliberation) by reconciling desires and intentions.
– Provide an explicit and an as detailed as possible representation of agents' mental state. Agents utilize a comprehensive set of mental attitudes based on the SharedPlans model for cooperation.

Key points for the evolution of ICAGENT towards our aim are the following:

– Specification of the multi-role recipe structure
– Extension of agents' reasoning tasks and plan management abilities for deliberating and reacting in a collaborative setting by exploiting multi-role recipes.
– Exploitation of roles' interdependencies for effective agents' coordination and communication.

As figure 2 shows, the ICAGENT overall architecture comprises two units: the Deliberation Control Unit (DCU) and the Plan Elaboration and Realization Control Unit (PERCU). These units, as well as the perception module consult and update agent's knowledge base.

Based on this architecture, an agent monitors its environment via the perception module and updates its beliefs about the environment. The term environment denotes the external, physical or simulated, environment as well as the mental attitudes of other agents acting in the environment. Although the perception module can be quite sophisticated, involving planning and multi-modal perception, this paper assumes that the agent, somehow, is aware of everything occurring in its environment.

The agent recognizes situations and forms desires to perform actions. While the agent may have many and possibly conflicting desires, depending on the situation at a specific time point, it must decide which action to pursue and whether it shall elaborate its plan towards that action reactively or deliberatively. Depending on whether the agent reacts or deliberates, it commits to the corresponding action, or it reconciles its desires with its intentions, reasoning about the relative strength of conflicting actions, about the strength of its commitments and its desires, and about the overall context of action.

As already noted, each action is realized by one or more alternative recipes. During plan formation, the agent selects relevant recipes, tests for their applicability and adds them in the overall plan. In this way, the agent constructs a hierarchical plan. This plan, augmented with constraints that must hold during plan formation and execution (e.g. preconditions of recipes) is referred as the context of action. Elaborating a plan, the agent reaches basic-level actions (i.e., actions that may be performed directly in the environment) and decides whether these actions shall be performed at the current time point, interleaving

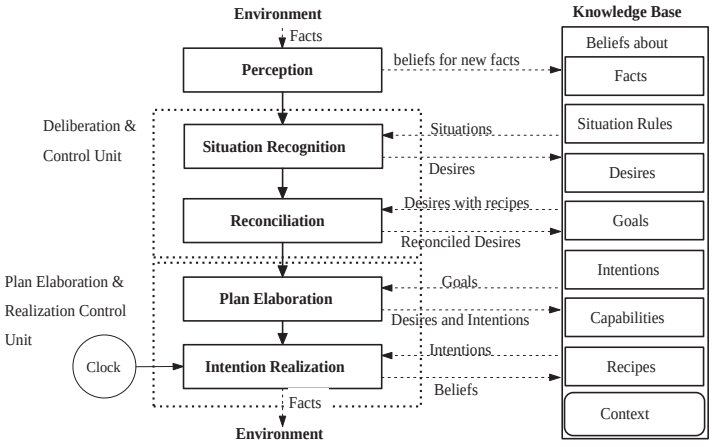

Fig. 2. The ICAGENT Architecture

planning with execution, or whether their execution shall be postponed until it has completed the corresponding part of the plan.

The structure and content of the resources depicted in figure 2, as well as the function of the individual modules are described in detail in [8].

4.1 The recipe structure

A recipe has the structure: `rec(action,recId,mntlCond,mode,type(recType, interleave),capConstr,cConstr,actionList,effects)` where:

- `action`, describes the action that the recipe realizes and has the form: `actionName(time,actionArgument`$_1$`,actionArgument`$_2$`,...)` where `time` is the time point that the action will be performed. Action arguments are either constants or variables. Variables are instantiated by checking the `mntlCond` constituent of the recipe, or during recipe selection.
- `recId` is an id for the recipe.
- `mntlCond` is a list of logical propositions. Each proposition specifies conditions for a certain role and combines mental attitudes using and, or and not logical connectives. The general form of the mental condition is: `[roleid`$_1$`:logicalProp`$_1$`,roleid`$_2$`:logicalProp`$_2$`,...]`, where `roleid`$_i$ is a list of role names.
- `mode` has the form: `[roleid`$_1$`:mode`$_1$`,roleid`$_2$`:mode`$_2$`,...]` where `mode`$_i$ has the form: `mode(BMntlCond,Behaviour)`. `BMntlCond` is a logical proposition and `Behaviour` is a variable that is instantiated to a, possibly empty, list of check directives that involve features that must be checked during reconciliation. It is this feature that enables each agent to balance between reactive and deliberative behaviour.

- `recType` is used to distinguish among domain recipes and communication protocols.
- `interleave` is a true/false variable. If true, then the agent interleaves planning with execution. Otherwise, the agent constructs the full plan for the corresponding action (either reactively or deliberatively) and executes the resulting plan afterwards. This argument is specified either by the agent developer or it is instantiated by checking the mental conditions of the recipe.
- `capConstr` stands for capability constraints and is a list that represents constraints that should hold during performance/reconciliation of roles. The list has the following form: $[\texttt{roleid}_1\texttt{:capConstr}_1\texttt{,roleid}_2\texttt{:capConstr}_2\texttt{,...}]$. $\texttt{capConstr}_i$ comprises a logical proposition that combines agent mental attitudes using and, or and not logical connectives.
- `cConstr` stands for contextual constraints. This is a list that represents constraints that should be maintained when the agent plans deliberatively towards actions under some specified role. The list has the following form: $[\texttt{roleid}_1\texttt{:conConstr}_1\texttt{,roleid}_2\texttt{:conConstr}_2\texttt{,...}]$. $\texttt{conConstr}_i$ is a logical proposition that combines agent mental attitudes using and, or and not logical connectives. Capability and contextual constraints constitute the preconditions of some recipe role and determine the applicability of that role and consequently the applicability of the recipe.
- `actionList` is a list that specifies the sequence of sub-actions that each role must perform. If this list is empty, then the action is a basic level one. The form of the action list is as follows: $[\texttt{roleid}_1\texttt{:action}_1\texttt{,roleid}_2\texttt{:action}_2\texttt{,...}]$. Agents that have committed to a role must perform the appropriate actions.
- `effects` is a list of facts that each agent that performs a role in the context of the recipe shall believe, when the plan towards `action` has been performed successfully.

4.2 Recipe's example and exploitation

Below is the definition of a recipe for the MAT, which is used in order one agent to get the tile that is closer to it. This recipe comprises two roles, `carrier` and `loader`.

The first role specifies that someone must get the tile, while the second role specifies that someone must load the tile to the first one. Mental conditions specify that `carrier` must recognize the closest tile that is not reserved by some other agent, while `loader` must check and confirm the existence of that tile. As far as the capabilities that agents should have are concerned, `loader` must be able to lift the desired tile and `carrier` must be able to carry it. Concerning context constraints, both roles should check if there is plenty of time in order to get to the tile's position. This is done by calculating the time needed to go to the position that the tile is located, comparing this with the lifetime of the tile. Concerning actions that must be performed, `carrier` must reserve the tile, both must go to the position of the tile, and finally, `loader` must load the tile to `carrier`.

```
rec( get_tile(_T),
    get_tile,
    [[carrier]: bel(agent_name(Agent1)) and
                bel(tile(TileId,Position,TTL,Type)) and
                not( bel(tile(TileId2,Position2,_,_)) and
                    TileId \== TileId2 and
                    closer_than(Position2,Position) ) and
                not bel(reserved(_SomeAgent,TileId)),
    [loader]: bel(agent_name(Agent2)) and
            bel(tile(TileId,Position,TTL,Type)) ],
    [[carrier,loader]: behaviour(reconc(...,ReconcDirect),failure(...)) ],
    type(domain([Agent1,Agent2]),true),
    [[carrier]: null, [loader]: bel(cap(Agent2,lift,tile_type(Type))) ],
    [[carrier]: bel(tile(TileId,Position,TTL,Type)) and
                calc_path(Position,_Path,PathCost1) and TTL > PathCost1,
    [loader]: bel(tile(TileId,Position,TTL,Type)) and
            calc_path(Position,_Path,PathCost2) and TTL > PathCost2 ],
    [[carrier]:         reserve(t(now,_),tile,TileId),
    [carrier,loader]: move_to(t(now,_),Position),
    [loader]:         load_tile(t(now,_),Agent1)],
    [[carrier,loader]: [not tile(TileId,Position,TTL,Type),
                        location(Agent1,Position),location(Agent2,Position)]]).
```

Fig. 3. The get_tile multi-role recipe.

This recipe explicitly specifies the actions that an individual or a team must undertake for performing action "get_tile". This is done by specifying the sequence of actions that must be performed, and the roles that should perform these actions. Consequently, a recipe captures interdependencies between roles by means of actions' temporal relations, as well as by arguments that these actions share.

Agents decide whether they shall utilize the recipe as a single or as a multi-agent recipe. An agent can commit to one or more roles according to its mental state, capabilities and the overall context of action. In case more than one agents commit to both roles, then the recipe is considered to be a multi-agent recipe.

Assume that agent in 7F adopts the recipe shown in figure 3 in order to get tile 8C. Agent identifies that it is not able to undertake the role `loader` of that recipe and decides to broadcast a request to the other agents in the MAT chessboard. Let us assume that agents 2B and 5J are willing to adopt the role `loader` in order to help agent 7F. If done deliberatively, these agents reconcile their desire to adopt `loader` role with other desires and commitments that may hold. If done reactively, these agents, based on their capabilities, commit to role `loader`. In case both agents commit to role `loader`, then the agent that maximizes team performance is chosen, while the other one retracts its intention to perform this role. In case both agents fill this role, then for each action that this role shall perform, they shall find a recipe and plan further towards their shared objective. In this way the recipe allows agents to decide on the best way for filling roles and performing actions in an arbitrary level of detail, building dynamic agent organizations.

The major stages, steps, and flow of control for agents involved in collaborative activity are shown in figure 4: agents form desires, form teams of collaborators, select recipes and allocate roles, and finally execute their roles. When an agent achieves, or fails to achieve the role to which it has committed, then it must inform the agents that share the same context with it. This may drive the team to reallocate roles or to select another recipe. This may further lead

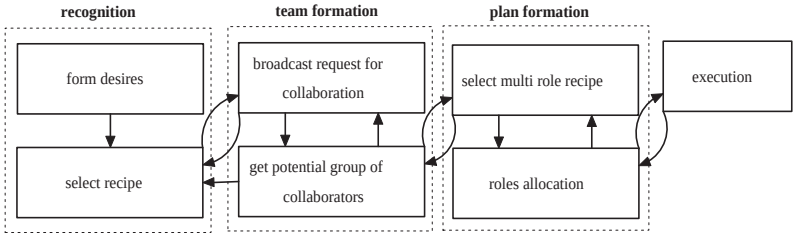

Fig. 4. The process of collaboration.

to the selection of another potential group of collaborators or even abandon the collaboration process. The organization of the group of collaborators through roles specifications provide the communication paths during cooperation. This is achieved via roles' interdependencies that can be inferred from the specification of multi-role recipes. These dependencies reduce the communication overhead, since agents communicate only with those collaborators on which their tasks depend (or depend by) and communicate only the necessary information for achieving their tasks. Therefore, roles provide an additional constraint that can be exploited by agents to decide on the messages and amount of information exchanged. Furthermore, roles specifications provide agents with necessary information to interpret other agents' actions in the context of their joined activity. This can also lead to the reduction of the communication overhead during collaboration. However, the ways that multi-role recipes affect communication is an issue of further work.

5 Concluding remarks

ICAGENT is an agent architecture implemented for real-time dynamic and unpredictable environments. This paper evolves ICAGENT by introducing multi-role recipes. Recipes constitute the know-how of agents and comprise one or more roles. Roles allow agents to decide on the actions that an individual or a team must undertake in the context of a recipe, facilitate agents to decide whether a recipe shall be utilised as a single or as a multi-agent recipe, and allow agents to decide on the best way for performing actions, planning in an arbitrary level of detail. Furthermore, being committed to common multi-agent recipes, agents coordinate their activities during planning and execution by capturing actions interdependencies, and communicate effectively by reducing communication overhead.

Further work concerns further evolution of the cooperation model introduced by multi-role recipes in both, the theoretical and the applications level. Moreover, we conjecture that using roles' recipes for modelling communication protocols, it is possible to catch the constraints and the dependencies among the interlocutors. This may enhance communication modelling and facilitate message interpretation.

References

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

[2] Eithan Ephrati, Martha E. Pollac, and Sigalit Ur. Deriving Multi-Agent Coordination through Filtering Strategies. In *Proceeding of the fourteenth International Joint Conference on Artificial Intelligence*, 1995.

[3] 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.

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

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

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

[7] 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.

[8] Vangelis Kourakos Mavromichalis and George A. Vouros. Balancing between Reactivity and Deliberation in the ICAGENT Framework. In Markus Hannebauer et. al., editor, *Balancing Reactivity and Social Deliberation in Multi-Agent Systems*, LNAI 2103, pages 53–75. 2001.

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

[10] 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.

[11] Luís Paulo Reis, Nuno Lau, and Eugénio Costa Oliveira. Situation Based Strategic Positioning for Coordinating a Team of Homogeneous Agents. In Markus Hannebauer et. al., editor, *Balancing Reactivity and Social Deliberation in Multi-Agent Systems*, LNAI 2103, pages 175–197. 2001.

[12] Charles Rich, Candace Sidner, and Neal Lesh. COLLAGEN: Applying Collaborative Discourse Theory to Human-Computer Interaction. *AI Magazine, Special Issue on Intelligent User Interfaces*, 2002. (to appear).

[13] 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.

[14] 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.

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

[16] Michael Wooldridge and Nicholas R. Jennings. The Cooperative Problem Solving Process. *Journal of Logic and Computation*, 9(4):563–592, 1999.