C$_O$MR$_O$S – A Multi-Agent Robot Architecture

Paul Levi, Michael Becht, Reinhard Lafrenz and Matthias Muscholl

Applied Computer Science – Image Understanding, Institute of Parallel and Distributed High-Performance Systems (IPVR), University of Stuttgart, Breitwiesenstr. 20-22, D-70565 Stuttgart, Germany.
E-mail: {levi, becht, lafrenz, mmuscholl}@informatik.uni-stuttgart.de

Abstract. In this paper we propose a refined version of the COMROS multi agent robot architecture. The concepts we use to define the operational principles of our architecture are autonomy of each functional module, task decomposition, and decentralized control. The structure of our architecture is defined by elementary agents which are connected by two types of networks. Furthermore, we describe how formation driving can be realized within the architecture.

Key Words. Multi-Agent System, Robot Architecture, Formation Driving, Robotics, Cooperation

1 Introduction

The paradigm of agents and multi agent systems is one of the most promising approaches to develop and control individual robots and teams of robots. In the last years various researchers published different agent-models which have been used in the field of robotics. Our approach is strongly influenced by biological systems. The concept we use to define the operational principles of our architecture are autonomy of each functional module, task decomposition and decentralized control. Therefore each module is defined by the types of information processing and acting it has to perform. Such a functional component is called sensor-actor cycle and is controlled by a decision unit which is defined through a set of roles it is committed to. We put both components together and call this unit elementary agent, where the sensor-actor cycle represents the body and the decision unit the head.

The structure of our architecture is defined by two types of networks which we suggest in order to connect the elementary agents. Decision networks bring all decision units together which are needed in order to perform a task. Data flow networks connect the bodies of elementary agents. Different tasks are performed by different decision networks which are running in parallel and not by creating new task specific agents as it is usually done.

In this paper we describe the concepts of our refined COMROS architecture (Levi et al., 1995) and give an comparison to other work, and an outlook on how we will establish our approach in a real system and try to verify our results by methods of synergetics.

2 Goals of COMROS

In the long run the COMROS project aims to develop a general purpose architecture for autonomous robots and teams of autonomous robots. The goals for the resulting robot architecture include predictability (computability) of distributed negotiation, reusability, dynamic configurability, fault tolerance, and safety. The suitability of the architecture will be evaluated through application to significant scenarios which are planned to be realized for different robot platforms.

As a short to mid term goal we decided to implement different realizations of formation driving as the first scenario for several reasons. There is a reasonable amount of existing work on this topic, which allows to compare our approach to others. Formation driving requires a variety of basic skills on the reflexive layer of the individual robots, where there is a need for cooperation, and it provides enough complexity to evaluate different strategies from more centralistic forms of control mechanisms to totally decentralized solutions. On the other hand formation driving is a basic task which is part of other more complex tasks like playing soccer (RoboCup) or collaborative transport.

3 System Architecture

The overall architecture is divided in three layers: strategical, tactical, and reflexive layer, reflecting different response times. Each level consists of several concurrent software components called elementary agents (EA) or autonomy cycles. Elementary agents are autonomous and are divided into a body (sensor-actor cycle) and a head (decision unit). The interaction between the decision units is based on dynamic configurable networks.

According to (Wettstein, 1993) we study the interaction patterns which can be established between a set of EAs, to perform a given task. An interaction pattern is described by several aspects: the service relation between agents, the extent of parallelism, the type of cooperation, the degree of synchronism, and the set of supporting organizational structures.

We distinguish two levels by which EAs interact with each other: the data flow between bodies and the cooperative decision making between the heads. The markedness of aspects of interaction patterns for the data flow are to some extent part of the concrete system architecture and therefore static, such as the organizational structure or the extent of parallelism given by the redundancy of sensors. The markedness of the other aspects is chosen at run time. The interaction patterns between EAs concerning cooperatively made decisions change dynamically depending on the active tasks.

The possible organizational structures and the quality of the service relation depend on the amount of available resources. Therefore, the scheduling of the resources is a crucial planning task, which has to coordinate those autonomous decisions of the EAs which affect the resource allocation.

3.1 Function and Structure of an Elementary Agent (EA)

One fundamental component of our architecture is the sensor-actor cycle as the body of an elementary agent. The idea of sensor-actor coupling resembles the fact that sensors provide a possibility for control and that actors should be able to provide information about the real state of the actions they are performing. The coupling of sensors and actors defines the autonomy domain of an elementary agent. Using the actor to improve the sensor results or vice versa is achieved through establishing a control loop between them. The algorithms used by *body* are defined by

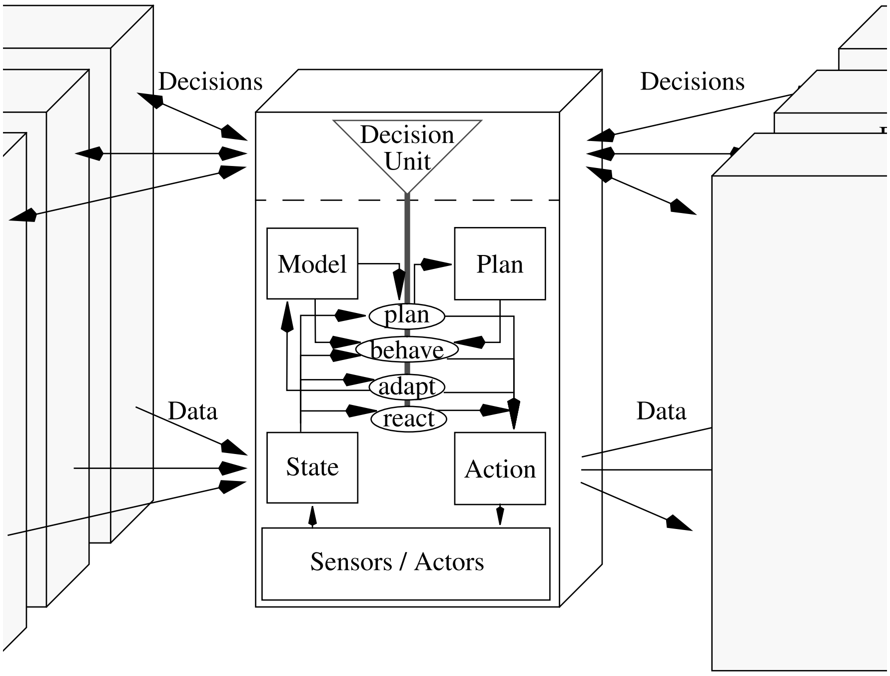

Figure 1 *An elementary agent.*

the autonomy of the sensor-actor cycles. This approach offers a uniform view of control mechanisms of different complexity ranging from rather simple ones like maintaining a given speed to advanced tasks like camera based object tracking.

Furthermore, the concept of sensor-actor coupling is not restricted to elementary tasks directly accessing the hardware but is applied on all levels of abstraction resulting in virtual sensor-actor cycles. These virtual sensor-actor cycles act through sending requests to other EAs and sense through receiving their results.

The internal structure of an EA is depicted in figure 1. As mentioned above the structure of an EA is composed of the sensor-actor cycle and the decision unit. The sensor-actor cycle is an abstract control loop and the decision unit controls the loop and manages the negotiations between the agents. The nodes of the S/A-cycle represent the data structures of world *models*, *plans*, *actions,* and *states*. The nodes are interconnected by a control loop which is composed of the four cycles *react*, *behave, adapt*, and *plan*. *React* is the shortest feedback between sensory states and actions. *Behave* is the model based feedback loop whereas *adapt* updates the structure of the model and *plan* creates new plans.

As a matter of fact different EAs may have intersecting world models, which means that parts of their models are subject to restrictions, which in turn results in a need for coordination. We use cooperation processes to coordinate the interdependent EAs.

3.2 Cooperation Processes

Each cooperation network is described by a cooperation process (Becht et al., 1997, Becht et al., 1998), which defines the phases, the actions, and the decentralized control structure of the cooperation.

3.2.1 Cooperation Model The underlying cooperation model consists of four components: control structure, role, action and object, whereas control structure is the central element. It specifies the overall processing by combining roles of the cooperation process, a set of actions, conditions concerning their invocation and the affected objects.

An action can involve a single role (local action) or a set of roles (cooperative action) which itself is described by a cooperation process. This results in a recursive definition of cooperation processes terminating in local actions.

3.2.2 Dynamics At design-time roles are considered as an abstraction of individual agents. At run time one or several roles can be contracted by an individual agent. By carrying out a role the agent fulfills the related behavior in the way the control structure permits it. The control structure decomposes the cooperative behavior in a series of phases on which each role performs either cooperative or local actions. During a phase each role publishes action-specific internal states to related roles. The combination of those internal states then triggers the roles to switch into the subsequent phases as specified in the control structure.

3.2.3 Data Flow Networks The focus of DFNs (Levi et al., 1995) is to describe the long-lasting connections between the bodies of EAs. A data flow network is determined by the compatibility of interfaces of the sensor-actor cycles. This determines the organizational structure of the robot agent, reflecting its built-in capabilities. Therefore the DFN defines the frame for the concrete markedness of interaction patterns, which can be established by autonomous or cooperative decisions of agents.

3.2.4 Decision Networks In order to achieve a task it is necessary to coordinate the individual decisions of the agents. As described above, in a complex scenario the autonomy domains of different agents cannot be treated independently. We therefore model the restrictions between the decision variables of the agents explicitly as a constraint network, in which each constraint has a related priority. In addition to real world restrictions, we use agent-specific constraints over the decision variables to inform the other agents about its utility.

Instead of connecting all agents in one huge decision network, we identify clusters of decision variables which are highly interdependent, thus forming a separate decision network. Decision variables which are contact points between two decision networks are modeled in both, leaving the task to the agent to coordinate the values. Such an agent is supposed to be a representative of the other decision network, handling the constraint network like its own utility constraints.

Based on the constraint network we design a cooperation process, in which the constraint propagation problem is solved cooperatively. We define a single role as a *member* of the decision network. Its task is to keep his constraints satisfied. As soon as it detects a conflict it first tries to adjust its own variables. If this fails, it searches for a solution where decision variables of other agents are adapted. If it can find such a solution, the agents come into a negotiation phase. Otherwise they have to relax some constraints, which is done by using a related priority mechanism.

3.3 Task Specific Resource Allocation

In order to achieve a flexible structure, the allocation of the resources defined by the data flow network of a certain robot is crucial. The quality of the allocation strategy determines whether different tasks can be run in parallel and how accurately a specific task is executed. The coordination of the resource allocations has to be done in advance and during run time through a configuration process. Run time adaption is necessary in the case of discrepancy between the assumptions at planning time and the reality. It has an interface to the task assignment (which is part of the strategical layer) and to other robots‘ configuration processes.

The result of a configuration process is a time dependent resource allocation plan together with the special characteristics of related restrictions. The DFNs and the individual decision units ensure that the required resources do not exceed the available capabilities based on the assumptions made at planning time. Nevertheless it cannot be guaranteed, that in every condition the planned resources are sufficient to perform the task. Therefore (Levi et al., 1995) have introduced a dynamic priorization mechanism to determine one or more masters which undertake the control of the system.

As for any decision process the possible realizations of the configuration process range from centralistic to democratic ones. In the decentralized approach favored by us, the configuration process is performed through a decision network on the tactical layer. To optimize the interaction between the tasks, priorities in resource assignment can be adapted.

The realization of the configuration process is open for later work, but it is part of an integrated design of the architecture.

4 Formation Driving as an Example Task

In section 3 we presented an architecture which can be used to implement a wide range of different tasks without redesigning the system. Many cooperative tasks like playing soccer or cooperative transport (Becht et al., 1997) need formation driving as a basic skill for a team of robots. In this chapter we describe how this specific task is incorporated in the presented architecture.

4.1 Requirements

Given a start and an end point for the team, the system has to negotiate how to reach the goal cooperatively. To be able to drive in a formation, one needs a figure (shape), and a trajectory for the whole team. Each team member has to know its position within the negotiated figure. Distances and angles between robots define the geometrical restrictions of the formation. One main task in formation driving is to keep these restrictions at any time. Of course, these restrictions are subject to change, e.g. while turning or driving in narrow corridors. Several realizations are possible depending on the amount of situative knowledge in each autonomous component.

Each robot has to know the trajectory of the formation in order to determine or negotiate its own trajectory and position within the group.

4.2 Elementary Agents (EA)

At the lowest level (with immediate access to the hardware) we find sensor-actor agents controlling the hardware and supplying sensor information in a uniform data structure. Among these agents are ultrasonic EA, laserscanner EA, and robot control EA (see figure 2). The fusion of sensor data is provided in EAs, which combine two or more physical sensors to a virtual sensor. The virtual distance sensor in this example combines the sensory information provided by laserscanner and ultrasonic

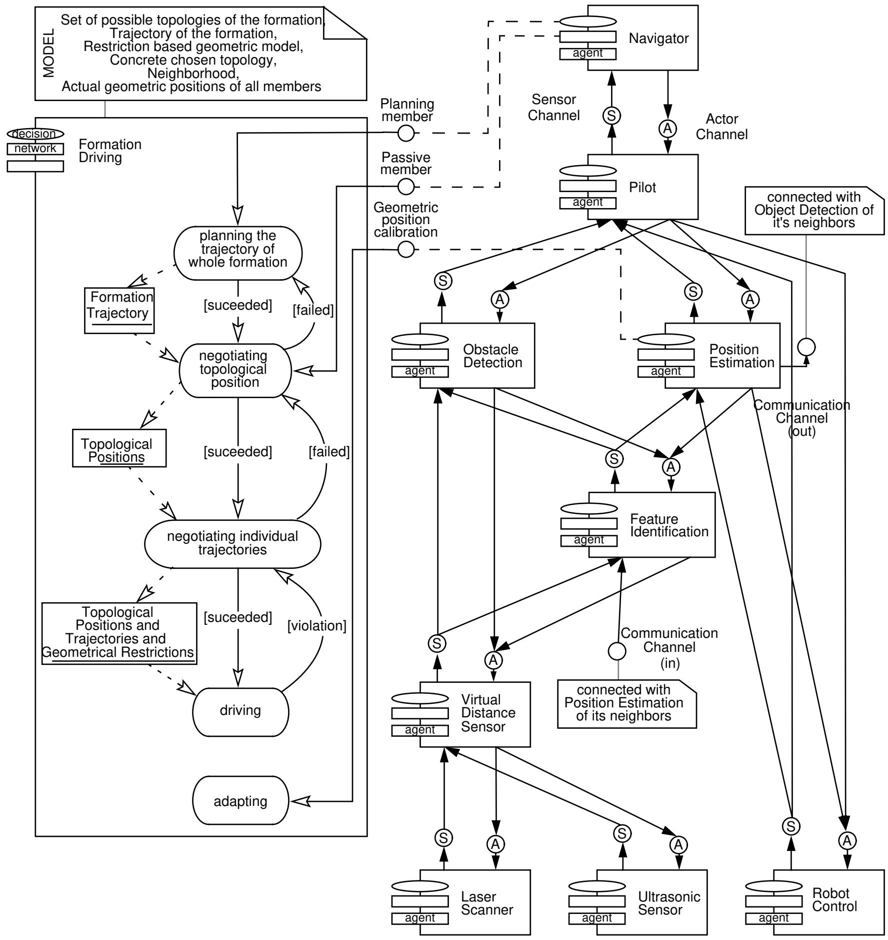

Figure 2 *Decision and data flow network for the formation driving task (depicted in unified modeling language-notation).*

EAs. To keep the formation in shape, the pilot needs information about the environment and the vehicle‘s position. Therefore it is connected to obstacle detection and position estimation EAs. Both of them use the information of the feature identification EA. Feature identification relies on the distance sensor and on communication to the position estimation EAs of other robots.

On a higher level, but still in the reflexive layer, we find the pilot, which is responsible for path following. It is given the next few stretches and geometrical restrictions. It is responsible to follow the given trajectory as good as possible, to avoid obstacles, get back on the trajectory, and to report problems to its navigator.

In order to keep the robots autonomous and hence not to introduce a central instance, we use existing agents, e.g. navigator and position estimation instead of establishing an new agent to control the group behavior. The formation driving behavior is achieved through interconnecting those agents by the formation driving decision network.

4.3 The Data Flow Network

An edge between elementary agents in the DFN consists of a sensor and an actor channel. The interaction pattern, which is often used in data flow networks is a client-server relation. Control information and model data are given by the client through the actor channel, whereas (virtual) sensory data and status information are reported by the server via the sensor channel.

For example, the interaction pattern between navigator and pilot is characterized by clear distinction between their responsibilities (in contrast to the relation between obstacle detection and feature identification). The navigator gives the planned trajectory together with geometrical and temporal restrictions to the pilot. The pilot continuously reports the actually driven trajectory and other relevant information like ongoing collision avoidance maneuvers to the navigator. The driven trajectory may differ from the one adviced by the navigator. With this information, the navigator is able to react in time or to negotiate the next trajectory stretch with other navigators.

4.4 Decision Networks

The roles of the decision network for formation driving are: *planning member*, *passive member*, and *geometric position calibration* (see figure 2).

Initially, the task of the planning members is to cooperatively plan the trajectory of the whole formation. This includes to choose a formation figure from a set of possible topologies of the formation. Due to environmental limitations it could be necessary to change the figure during the formation.

After the planning phase has been completed successfully, all members of the formation negotiate the assignment of robots to topological positions. This is done for all figures planned in the previous phase. Therefore each member has a model of his capabilities as well as the requirements of each topological position.

In the next phase each member calculates its trajectory relative to the trajectory of the formation. Conflicts may occur in sections, where the formation has to be restructured. As the result of this phase each individual trajectory is attributed with geometric restrictions to neighbors in the formation.

The result is used in the driving phase, which either terminates by reaching the end point or fails due to a restriction violation. If one phase fails, the roles will backtrack to the previous phase.

During driving, the roles *geometric position calibration* correct the discrepancies which buildup through drift or inaccuracies in the odometry.

The decision network does not determine the assignment of roles to elementary agents. Each role specifies the skills an agent must have. An agent that matches the requirements commits itself to a role. In the example the roles *planning member* in and *passive member* require navigator capabilities. Agents knowing the start and the end point and have according planning capabilities are able to take the planning member role. In military domains only the formation leader will have this knowledge. In a traffic scenario, where each individual vehicle has its own start and end point every navigator will match this criteria, ending up with a convoy driving a common route.

5 Comparison to other Architectures

(Laengle and Lueth, 1996) proposed a multi-agent structure, where an agent is decomposed into a head and a body. The head is responsible for task acquisition (planning head) and reservation of resources (acting head). In cooperative tasks a special agent fulfills the work of two or more heads, and controls several bodies. In contrast, our architecture is based on decentralism and negotiation. Furthermore, resources are controlled by decision networks in which EAs are cooperatively responsible for managing parallel access.

In (Mataric, 1997) a behavior-based architecture is used, where basic behaviors are directly or temporally combined to form higher-level cooperative behaviors. In our architecture, each behavior is reflected in a decision network.

In the ALLIANCE architecture (Parker, 1994), behavior sets represent high-level functions. For the decision which behavior set is to be activated, motivational behaviors, like impatience, and cross-inhibitions between them are implemented. Both behaviors could be modeled by specific decision networks.

The article (Cao et al., 1997) supplies a good overview of the recent cooperative robotics research. Further, (Hexmoor, 1997) deals with various aspects of software architecture for hardware agents.

6 Outlook

The proposed architecture is planned to be realized on the three COMROS robots which are controlled by a workstation cluster as described in (Bayer et al., 1995) and on a mini-robot platform.

Another focus of the research activities is to use methods of synergetics to identify interaction patterns and to examine their stability. This allows to detect and distinguish stable from unstable, periodic from aperiodic, and chaotic from not chaotic states. Furthermore it is possible to define critical boundaries of parameter values. This allows the identification of structural state changes (e.g. bifurcations). For more details see (Levi et al., 1998).

7 REFERENCES

Bayer, H., Th. Bräunl, A. Rausch, M. Sommeraul, and P. Levi. (1995). *Autonomous Vehicle Control by Remote Computer*. In U. Rembold, R. Dillmann, L.O. Hertzberger, and T. Kanade, editors, Intelligent Autonomous Systems IAS-4, pages 158 - 165, Amsterdam, Oxford, Washington DC, 1995. University of Karlsruhe, IOS Press.

Becht, M., M. Muscholl, and P. Levi. (1997). *Ein Framework für Kooperationsverfahren zwischen Roboteragenten*. In P. Levi, Th. Bräunl, and N. Oswald, editors, *Autonome Mobile Systeme 1997*, Informatik aktuell, Berlin u. a., Springer

Becht, M., M. Muscholl, and P. Levi. (1998). *Transformable Multi-Agent Systems: A specification language for cooperation processes*. In Proceedings of the World Automation Congress (WAC), Sixth International Symposium on Manufacturing with Applications (ISOMA), 1998.

Cao, Y.U., A.S. Fukunaga and A.B. Khang (1997). *Cooperative Mobile Robotics: Antecendents and Directions*. Autonomous Robots **4**, 7-27

Hexmoor, H. (ed.) (1997). *Journal of Experimental & Theoretical Artificial Intelligence,* Vol. 9, Nos. 2-3

Levi, P., M. Muscholl, and Th. Bräunl. (1995). *Cooperative Mobile Robots Stuttgart: Architecture and Tasks*. In U. Rembold, R. Dillmann, L.O. Hertzberger, and T. Kanade, editors, Intelligent Autonomous Systems IAS-4, pages 310 - 317, Amsterdam, Oxford, Washington DC, 1995. University of Karlsruhe, IOS Press.

Levi, P., M. Schanz, and V. Avrutin. (1998). *Modelling and analysis of a simple manufacturing-oriented multi-agent system*. In Intelligent Autonomous Systems 5, 1998

Längle, Th., T. Lüth and M. Hesse (1996). *Verklemmungsfreie Agentenkooperation in verteilten Robotersystemen*. In *Autonome Mobile Systeme 1996,* Springer Informatik Aktuell

Mataric, M. (1997). *Bahavior-based Control: Examples from Navigation, Learning, and Group Behavior*. *Journal of Experimental and Theoretical AI* Vol. 9, Nos. 2-3, pp. 323-336

Parker, L.E. (1994). *Heterogeneous Multi-Robot Cooperation*. PhD Thesis, MIT

Wettstein H. (1993). *Systemarchitektur*. Hanser Studienbücher der Informatik. Hanser, München - Wien, 1993.