MAS2CAR ARCHITECTURE
Multi-agent System to Control and Coordinate teAmworking Robots

Mehdi Mouad[1], Lounis Adouane[2], Pierre Schmitt[1], Djamel Khadraoui[1] and Philippe Martinet[2]
[1]*Centre de Rechere Public Henri Tudor, Luxembourg, Luxembourg*
[2]*Laboratoire des Sciences et Matériaux pour l'Electronique et d'Automatique, Université Blaise Pascal Clermont-Ferrand, France*

Keywords: Multi-robots system, Multi-agent system, Organizational model, Robots cooperation, Robots control architectures, Hybrid control.

Abstract: This paper aims to present the Multi-Agent System to Control and Coordinate teAmworking Robots (MAS2CAR), a new architecture to control a group of coordinated autonomous robots in unstructured environments. MAS2CAR covers two main layers: (i) the Control Layer and we focus on (ii) the Coordination Layer. The control module is responsible for a part of the decision making process while taking into account robot's structural constraints. Despite this autonomy possibility, the Coordination Layer manages the robots in order to bring cooperative behavior and to allow team-work. In this paper we present a scenario validating our approach based upon the multi-agent systems (MAS). Thanks to its reliability we have chosen the $\mathcal{M}$ OISE^{Inst} organizational model and we will present how it can be used for this use-case. Moreover, regarding to the implementation part, we have retained $\mathcal{U}$TOPIA, a framework which automatically build a MAS thanks to a $\mathcal{M}$ OISE^{Inst} specification.

1 INTRODUCTION

A multi-robot system can be defined as a set of robots operating in the same work space. Given some robotics task specified by a designer, a multiple-robot system can benefit from cooperative behavior if, due to some cooperation or coordination mechanism, there is an increase in the total utility of the system.

In the case of mobile robotics, the need to operate in increasingly complex and unstructured environments, and at the same time reduce costs to a minimum in terms of time, power, etc.), raise the development of autonomous robots. The ultimate goal is the capability of achieving coordinated movements and of carrying out tasks that usually require human assistance. This need for autonomy requires from the robot a certain capacity of being able at any moment to assess both its state and its environment that are usually combined with different other robots states as well as with its mission requirements in order to make coherent control decisions. If we consider navigation aspects, autonomous mobile robots are usually embedded with sensors/actuators according to the mission to be performed. This complexity induces major challenges both at the development of robotics control architecture system but also at the design of navigation software.

Indeed, an autonomous mobile robot has to carry out a set of sensors/actuators dedicated to its own navigation and another sensors set that can change according to the mission to be performed. Therefore the navigation software developed for these vehicles become complex and requires a design methodology.

This paper aims at presenting MAS2CAR[1], an architecture model for multi-mobile robots based on Multi-Agent System (MAS) coordination. This paper is organized as follow: in section 2 we make an overview of the related works in the areas of multi-robots and MAS, in 3 we introduce our architecture model focusing on the tree main layers of our model: Physical Layer, Control Layer and Coordination Layer. Subsequently, we focus on the Coordination Layer and we explain how the MAS have been used. More particularly, and we describe the Organization Specification (OS) in section 4.

[1]Multi-Agent System to Control and Coordinate teAmworking Robots (MAS2CAR).

Mouad M., Adouane L., Schmitt P., Khadraoui D. and Martinet P..
MAS2CAR ARCHITECTURE - Multi-agent System to Control and Coordinate teAmworking Robots.
DOI: 10.5220/0003649904510456
In *Proceedings of the 8th International Conference on Informatics in Control, Automation and Robotics* (MORAS-2011), pages 451-456
ISBN: 978-989-8425-75-1

2 STATE OF THE ART

Robotics software is now one of essential part of robotics system development. Therefore, software architectures design methods and concepts, are mainly made to enhance evolutionary, modularity... and to avoid redesign costs. The last years have seen active research in the field of distributed robotics, and in the development of architectures for multi-robot coordination. These architectures have focused on providing different capabilities to the group of robots. For instance, ALLIANCE (Parker, 1998), a behavior-based software architecture, has focused on fault tolerant cooperative control. In Morrow and Khosla (Morrow et al., 1997), robot skills are expressed as finite state machines (FSM).

The coordination of robots for large-scale assembly has been considered in Simmons et al. (Simmons et al., 2000). Klavins and Koditschek (Klavins et al., 2000) have presented tools for composing hybrid control programs for a class of distributed robotics systems. This approach assumes that a palette of controllers, each one implements a bihavior. These controllers, i.e, robot behaviors, are sequentially composed using the techniques introduced in Burridge, Rizzi, and Koditschek (Burridge et al., 1999). These ideas are applied to the design of assembly tasks as found in automated factories.

Control software architecture design approaches are usually classified into three main categories:

- Reactive v.s. Cognitive (deliberative) architectures, many modules connects several inputs sensors/actuators, each module implements a behavior. These behaviors are called “reactive” because they provide an immediate output of an input value, and cognitive otherwise.
- Hierarchical v.s. Non-Hierarchical architectures, the hierarchical architectures are built in several levels, usually three. Decisions are taken in the higher level; the intermediate level is dedicated to control and supervision. The low level deals with all periodical treatment related to the instrumentation, such as actuator control or measuring instrument management (Lumia et al., 1990).
- Hybrid architectures are a mix of the two previous ones. Usually these are structured in three layers: the deliberative layer, based on planning, the control execution layer and a functional reactive layer. It’s in the same time reactive with a cognitive level (planning for example).

To bring coordination into a multi robotics system we can distinguish centralized approaches from distributed ones.

- A centralized system has a robot (leader) in charge of organizing the work of the other robots. The leader is involved in the decision process for the whole team, while the other members can act only according to the leader’s directions.
- In contrast, a distributed system is composed of robots that are completely autonomous in the decision process with respect to each other; there is no leader in such cases.

Among multi-agent based Robotics Development Environments (RDE), OROCOS architecture (Bruyninckx, 2001) is a modular framework capable of controling multi-robot systems providing an environment for implementation of real-time control systems with various abstraction levels for hardware device drivers.

ARTIS architecture (Botti et al., 1999) allows developing agents working in hard real-time environments. Using an off-line analysis of the specification, the architecture performs the execution of the entire system. The Agents allows the self-adaptation in the changing environments, by executing tasks autonomously.

IDEA architecture (Intelligent Distributed Execution Architecture) (Muscettola et al., 2002) based on a multi-agent system to control multi-robot systems. Where each agent can be a functional module, a planner, a diagnostic system, ... The operation of agents is based on the “procedure” and “token”. IDEA agents can communicate and monitor each other. The database is partitioned online of time (timelines), each representing the temporal evolution of a sub-system property.

ARTIS and IDEA architectures are a very interesting architectures, and both describes one agent architecture. When an ARTIS agent is applied to robot, it contains sensor/actuator modules, control modules for real time execution and a reflex layer for critical events, its in-agents are dedicated to the different behaviors such as localization, trajectory planner, radio communication, obstacle avoidance... And IDEA is a multi agent framework for planning and execution for agents, it’s composed of three layers: Token and procedures, communication wrapper, and a virtual machine which integrates planning as the reasoning module at the core of the execution engine, the interplaying of its different modules (the domain model, the plan database, the plan runner and the reactive planner) provides the basis for agents autonomy. both have a good coordination level if we consider them in a multi-robot context, but it does not rely on organizational rules to build agents, which is the case of our architecture, our agents are built through an organization, the organizational model takes into account all the tasks and constraints, and on this basis we build

our agents thanks to $\mathcal{U}$TOPIA .

There is an other multi-agent Hybrid architecture (Fierro and Das, 2002) which describe a high-level language with formal semantics, to describe multi-agent networked robotics systems. This architecture allows the development of complex multi-robot behavior via: hierarchical and sequential composition of control; estimation modes, and parallel composition of agents.

This architecture is the closest given into our model. Infact it's organized at the highest level of the hierarchy, in two interacting agents: a coordination agent and a robot-group agent, which approximately represent in our architecture: $\mathcal{M}$OISE^{Inst} the organizational model and $\mathcal{U}$TOPIA the MAS instantiation middelware. The coordination agent reduces to the specification of communication channels between robot agents, and the specification of parameters for transitions and the instantiation of each agent within the robot-group agent.

A MAS particularly meet the underlaying needs of supervision and coordination of multiple and mobile robots. For that, we have to associate an agent to each physical robot and model each robot as an agent in the MAS model.

Among the MAS models we have chosen the Electronic Institution (Esteva, 2003). Indeed, this organizational model allows to express cooperation schemes defined by the user with an Organization Modeling Language such as for instance $\mathcal{M}$OISE^{+} (Hübner et al., 2002), and OMNI (Dignum et al., 2004). The aim of these services is to constraint and supervise agent's actions and interactions in order to achieve some global goals. We call those explicit cooperation schemes OS.

To summarize our different ideas, we state to develop a hybrid architecture which takes into account: the advantages of Reactive and Hierarchical architectures, to obtain more efficient reaction in different aspects, such as having good level of data processing while minimizing the reaction time, allows coordination and permits hybrid distributed / centralized aspect. This architecture is dedicated to multi-robot systems with a high degree of coordination between autonomous robots. The main originality of MAS2CAR is the used coordination method and the challenge is to implement an organizational model for robotic agent with all the physical and automatical constraints to obtain a more powerful multi-robot coordination, and apply it on real robots.

3 MAS2CAR'S GLOBAL MODEL

In our architecture model:

- We focus on nonholonomic homogeneous[2] robots;
- The environment in which robots evolve is partially known but we also consider the possibility of encounter unexpected obstacles.

3.1 Overview

Elaborating an innovative architecture to control a group of coordinated autonomous vehicles in unstructured environments, have to take into account different aspects. That is why our model is composed by three main layers for every robot (cf. Figure 1):

1. **The Physical Layer** is composed of multiple sensors/actuators existing on the robot.
2. **The Control Layer** allows us the "basic" goals modules such as planning, re-planning, reactive-mode and the sensors/actuators management.
3. **The Coordination Layer** is dedicated to more complex abstract or social goals. Basically, this level is represented by an agent.

The originality of this architecture is to use Electronic Institution MAS model to bring cooperation, coordination and intelligence at both individual and social levels.

As shown in Figure 1, the Control Layer makes the link between the Physical Layer and the Coordination Layer (agent). It manages the sensors and actuators in order to serve abstract commands for every actuator and events from every sensor.

For instance, if the Coordination Layer decide to reach a target, the Control Layer have to exactly calculate the best trajectory to reach this objective, taking into account the robot's structural constraints: the non-holonomy, avoiding the set points discontinuities, the limitation of the rotational torques, etc. The robot must avoid also the known obstacles on the path. We have choosen the Potential Fields method to plan the robot trajectory, and the Orbital Obstacle Avoidance Algorithm (Adouane, 2009) for unexpected obstacles.

3.2 Properties

3.2.1 Communication

One of the main objects of study in multi-robot systems research is the communication or interaction be-

[2]A robot set is homogeneous if the capabilities of the individual robots are identical and heterogeneous otherwise.

tween the robots. Three main communication structures are often used (Werfel and Nagpal, 2006).

- First is communication or interaction via environment: this occurs when the environment itself is the communication medium.
- Another typical structure is the interaction via sensing: this refers to local interactions that occur between agents as a result of them sensing one another.
- Last is interaction via communications: this involves explicit communication with other agents, by either directed or broadcast intentional messages.

The most appropriate in the MAS environments is this last one, thanks to its directed and broadcast intentional messages, in our model we use it with a communication interface listening on every robots. Thanks to this, an agent of the Coordination Layer can connect to its associated Control Layer in two different ways: (i) Locally if the robot has the required characteristics to embed the agent directly; (ii) Remotely through a robot wireless interface.

3.2.2 Abstraction

Every agent connected to the Control Layer's interface can send and receive messages. The Control Layer behave as a middleware which receive commands for actuators and send events from sensors.

As an abstract layer for the hardware, this Control Layer design permits multi-heterogeneous robots, as the Control Layer can be adapted to different hardware while serving the same middleware.

For these reasons, *this architecture allows (i) Robot-independent Coordination Layer implementation (ii) platform-independent programming languages to implement the MAS and finally (iii) more powerful and reactive Coordination Layer.*

The Control Layer (cf. Figure 1) take basic decisions such as determining a trajectory or avoiding an unexpected object allowing the Coordination Layer to focus on more complex tasks or social behaviors.

This paper focus on the Coordination Layer presented in the next sections.

4 MAS2CAR'S ORGANIZATION SPECIFICATION

In this part we will describe an Institution-oriented approach to model our MAS. We will focus on giving the possibility for agents to reach a targeted point (objective) according to a partially known environment in

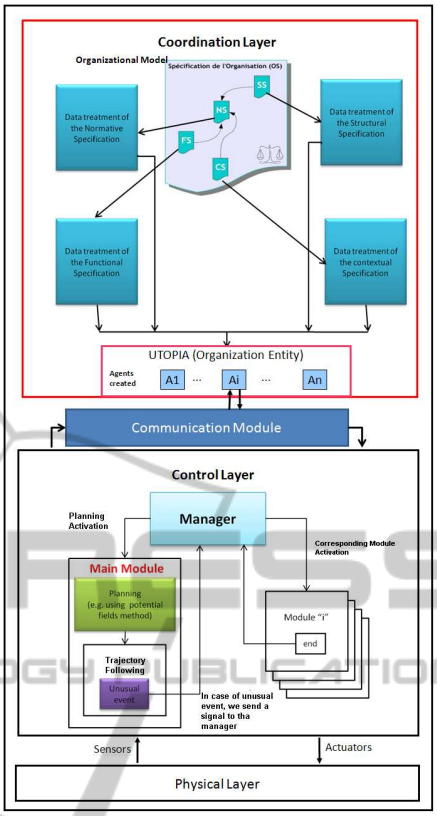

Figure 1: Architecture of a robot at the individual scope (Mouad et al., 2009).

a planning-based mode, we also model a possible reactive mode to face unexpected events.

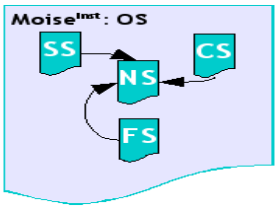

Figure 2: $\mathcal{M}$ OISE^{Inst}, a normative Organization Specification model, and its different specifications.

4.1 Structural Specification

Define a role (r) per robot: $rRobot\{i\}$ with $i = \{1..n\}$ and n the total number of robots. All these roles have a cardinality of 1. By this way one and only one agent is associated to each robot. In addition of the robot's roles, we have defined a role for a Supervisor, *rSupervisor*, which have the duty to manage the others agent with a global point of view.

Thus our architecture is a distributed architecture, and the MAS (Coordination Layer of our architecture,

cf. figure 1) is mainly composed by the agents representing an associated agent's Control Layer and by a supervisor, in each robot. In consequence we obtain a high level of coordination and cooperation for our group of robots due to the connection between supervisors in each robot.

4.2 Contextual Specification

A contextual specification can be seen as a recursive transition-graph where states are called *contexts* and set of contexts are called *scenes* (s). We have one scene for the supervisor, *sSupervisor*, and one for each robot: *sRobot*$\{i\}$ (cf. table 1). Theses scenes includes specific contexts influencing the behavior of the robots:

- Planning mode associated to the **initial** context *planningMode*, used as often as possible by the robots to compute their trajectories and reach their goals.
- Reactive mode associated to the context *reactiveMode*, triggered when an unexpected obstacle is detected in order to avoid it.

Thanks to transitions, we can switch the mode of each robots. Indeed, the organization is $n+1$ different contexts **at the same time** (n agents and 1 supervisor). At the beginning, we have: *sSupervisor/Active* and *sRobot*$\{i\}$*/planningMode/Active* (the agent i activate the Planning Mode) $i = \{1..n\}$. If due to unexpected obstacle, the supervisor send the transition *AO5* (which means "Avoid Obstacle" into the scene *sRobot*$\{5\}$), the contexts turn to be *sSupervisor/Active*, *sRobot5/reactiveMode/Active* and *sRobot*$\{i\}$*/planningMode* with $i = \{1..n\}$, $i \neq 5$.

4.3 Functional Specification

Here, we have specified *goals* and set of goals (*missions*) for the robots and the supervisor (cf. table 1). The goals are executed in a specific order according to three modalities:

- Sequence ($\rightarrow$): $g_1 \rightarrow g_2 \rightarrow ... \rightarrow g_{p-1} \rightarrow g_p$ the goal g_p have to be realized before g_{p-1}, ... , g_2 have to be realized before g_1.
- Parallelism ($\|$): $r \parallel \{g_1, g_2, ..., g_p\}$: the goals g_1, g_2, ... , g_p have to be realized in parallel after realizing r.
- Choice ($\pitchfork$): $r \pitchfork \{g_1, g_2, ..., g_p\}$ one and only one $g_{i \in [1..p]}$ have to be realized after realizing r.

We have three missions:

mSupervisor = *gRoot* $\|$ $\{gSupervisor, gCollisionSolver\}$

The supervisor have to do three set of actions in parallel: management of the messages and requests from the robots within *gSupervisor*, and detection / resolution of conflicts between robots within the goal *gCollisionSolver*.

mPlan = *gWaitSupervisor* $\rightarrow$ *gComputeTrajectory* $\rightarrow$ *gAskForPermission* $\rightarrow$ *gReachTarget*

First we execute *gWaitForSupervisor* (is the supervisor here and ready?) when done, we execute *gComputeTrajectory* wherein we ask Control Layer for a planning in order to reach a targeted point of the environment.

Then we execute *gAskForPermission* wherein we ask supervisor if the computed planning implies conflicts with other robots. As a result, we obtain constraints to avoid conflicts and we modify the planning to respect these constraints and robot characteristics.

Finally, the goal *gReachTarget* is executed and we send messages to the Control Layer in order to move according to the plan.

This goal run until the objective is reached or until it is interupted by messages coming from sensors (unexpected obstacle). In this case a transition (*AOi* where *i* is the identification of the robot) is sent to switch the robot into reactive mode.

mReact = *gAvoidObstacle* $\rightarrow$ *gReturnOnTrajectory*

mReact Is used to manage an unexpected obstacle detected by sensors. In *gAvoidObstacle* we move the robot according to data coming from sensors in order to avoid the obstacle. As soon as the obstacle is far enough we run *gReturnOnTrajectory* wherein we try to get back to the trajectory planified before. When it is done, we can leave the reactive mode and go back to the planning mode by sending a transition to change the context.

4.4 Normative Specification

In the normative specification (cf. table 1), we have a Norm for the supervisor *NSupervisor*, which forces the agent playing the role *rSupervisor* when the Institution is in the context *sSupervisor/Active* to do the mission *mSupervisor* described before.

For the *n* robots, we have two norms:

- *N*$\{i\}$*planningMode* forces the agent playing the role *rRobot*$\{i\}$ when the Institution is in the context *sRobot*$\{i\}$*/planningMode/Active* to do the mission *mPlan*.
- *N*$\{i\}$*reactiveMode* the same for role *rRobot*$\{i\}$ when the Institution is in the context

Table 1: Normative Specification glueing all three other specifications.

Normative Specification	Contextual Specification	Structural Specification	Functional Specification
NSupervisor	*sSupervisor/Active*	*rSupervisor*	*mSupervisor:* - Supervisor (recieve plans...). - Collision Solver (conflicts detection/resolution).
N{i}PlanningMode	*sRobot{i}/PlanningMode/Active*	*rRobot{i}*	*mPlan:* - Ask for Permission (from supervisor). - Compute Trajectory (plan trajectory using the potential fields method). - Wait Supervisor (permission given after the trajectory analysing). - Reach Target.
N{i}ReactiveMode	*sRobot{i}/ReactiveMode/Active*	*rRobot{i}*	*mReact:* - Avoid Obstacle. - Return to Trajectory.

sRobot{i}/reactiveMode/Active to do the mission *mReact*.

5 CONCLUSIONS

This paper describes the proposed control architecture with its three layers, while focusing on the Coordination Layer and have shown how a multi-agent system can be helpful to bring coordination and cooperation into a multiple heterogeneous robots set. To this end, the key point is twofold:

- Using an Organizational Model ($\mathcal{M}$ OISE^{Inst}) in order to describe the structure, goals and contexts in a normative Multi-Agent System.
- Taking the benefit of $\mathcal{U}$ TOPIA framework to handle the organization specification (OS) and translate it into a concrete MAS corresponding to the Coordination Layer throught our goal implementations executed by the robots.

This centralized / decentralized possibility is a key characteristic of MAS2CAR architecture as it allows to face every events while keeping a specification of global goals.

Future works will adress more sophesticated collaborative tasks, behaviors and team-work, such as the hierarchical fleet organization and dynamic changing environments...

REFERENCES

Adouane, L. (2009). Orbital obstacle avoidance algorithm for reliable and on-line mobile robot navigation. In *9th Conference on Autonomous Robot Systems and Competitions*.

Botti, V., Carrascosa, C., Julian, V., and Soler, J. (1999). Modelling agents in hard real-time environments. In *Lecture Notes in Computer Science*, volume 1847/1999, pages 83–78.

Bruyninckx, H. (2001). Open robot control software: The orocos project. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2001)*, volume 3, pages 2523–2528.

Burridge, Rizzi, R., A., Koditschek, and D. (1999). Sequential composition of dynamically dexterous robot behaviors. In *International Journal of Robotics Research*, volume 18(6), pages 534–555.

Dignum, V., Vazquez-Salceda, J., and Dignum, F. (2004). Omni: Introducing social structure, norms and ontologies into agent organizations. In *ProMAS International Workshop 2004*, New York, USA.

Esteva, M. (2003). Electronic institution: from specification to development. In *IIIA Ph.D. Monography, v19*.

Fierro, R. and Das, A. (2002). A framework and architecture for multi-robot coordination. In *The international Journal of Robotics Research*, volume 21, pages 977–995.

Hübner, J. F., Sichman, J. S., and Boissier, O. (2002). A model for the structural, functional, and deontic specification of organizations in multiagent systems. In *SBIA'02*, number 2507 in LNAI, pages 118–128. Springer.

Klavins, E., Koditschek, and D. (2000). A formalism for the composition of concurrent robot behaviors. In *IEEE Int. Conf. on Robotics and Automation*, volume 4, pages 3395–3402.

Lumia, R., Fiala, J., and Wavering, A. (1990). The nasrem robot control system and testbed. In *IEEE Journal of Robotics and Automation*, pages 20–26.

Morrow, J., and Khosla (1997). Manipulation task primitives for composing robot skills. In *IEEE Int. Conf. on Robotics and Automation*, volume 14, pages 3354–3359.

Mouad, M., Adouane, L., Khadraoui, D., and Martinet, P. (2009). Multi-agents system to control and coordinate teamworking robots (mas2car). In *7eme Journees Nationales de la Recherche en Robotique JNRR'09*.

Muscettola, N., Dorais, G., Fry, C., Levinson, R., and Plaunt, C. (October 2002). Idea: Planning at the core of autonomous reactive agents. In *Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space*.

Parker, L. (1998). Alliance: An architecture for fault tolerant multi-robot cooperation. In *IEEE Transactions on Robotics and Automation*, volume 14, pages 220–240.

Simmons, R., Singh, S., Hershberger, D., Ramos, J., and Smith (2000). Coordination of heterogeneous robots for large-scale assembly. In *ISER00, 7th Int. Symposium on Experimental Robotics*, pages 311–320.

Werfel, J. and Nagpal, R. (2006). Extended stigmergy in collective construction. In *IEEE Intelligent Systems*, volume 21, pages 20–28.