HAL
open science

Control Architecture for Cooperative Mobile Robots using Multi-Agent based Coordination Approach

Mehdi Mouad, Lounis Adouane, Pierre Schmitt, Djamel Khadraoui, Philippe Martinet

▶ **To cite this version:**

Mehdi Mouad, Lounis Adouane, Pierre Schmitt, Djamel Khadraoui, Philippe Martinet. Control Architecture for Cooperative Mobile Robots using Multi-Agent based Coordination Approach. 6th National Conference on Control Architectures of Robots, INRIA Grenoble Rhône-Alpes, May 2011, Grenoble, France. 15 p. inria-00599602

HAL Id: inria-00599602

https://inria.hal.science/inria-00599602

Submitted on 10 Jun 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Control Architecture for Cooperative Mobile Robots using Multi-Agent based Coordination Approach

Mehdi Mouad[1], Lounis Adouane[2], Pierre Schmitt[1], Djamel Khadraoui[1], and Philippe Martinet[2]

[1] Centre de Recherche Public Henri Tudor
Service Science and Innovation Dpt.
Luxembourg, G.D. of LUXEMBOURG
{firstname.lastname}@tudor.lu
http://www.tudor.lu

[2] LAboratoire des Sciences et Matériaux pour l'Electronique et d'Automatique,
CNRS Campus des Cézeaux
Aubiere, FRANCE
{lounis.adouane;philippe.martinet}@lasmea.univ-bpclermont.fr

Abstract. This paper is about a Multi-Agent based solution to control and coordinate team-working mobile robots moving in unstructured environments. Two main contributions are considered in our approach. The first contribution of this paper is about the Multi-Agents System to Control and Coordinate teAmworking Robots (MAS2CAR) architecture, a new architecture to control a group of coordinated autonomous robots in unstructured environments. MAS2CAR covers three main layers: (i) the Physical Layer (ii) the Control Layer and (iii) the Coordination Layer. The second contribution of this paper is about the multi-agent system (MAS) organisational models aiming to solve the key cooperation issues in the coordination layer, the software components designed based on $\mathcal{U}$TOPIA a MAS framework which automatically build software agents, thanks to a multi-agent based organisational model called $\mathcal{M}$OISE^{Inst} . We provide simulation results that exhibit robotics cooperative behavior related to our scenario, such as multi-robots navigation in presence of obstacles (including trajectory planning, and reactive aspects) via a hybrid control.

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 [1].

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[3], 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, we describe the Organization Specification (OS) in section 4, some MAS important implementation aspects in 5 and the simulation results in section 6.

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 [2], a behavior-based software architecture, has focused on fault tolerant cooperative control. In Morrow and Khosla [3], robot skills are expressed as finite state machines (FSM).

The coordination of robots for large-scale assembly has been considered in Simmons et al. [4]. Klavins and Koditschek [5] 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 [6]. 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 [7]:

[3] Multi-Agents System to Control and Coordinate teAmworking Robots (MAS2CAR)

- 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 [8] [9].
- 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 [10].
- Hybrid architectures are a mix of the two previous ones [7]. Usually these are structured in three layers: the deliberative layer, based on planning, the control execution layer and a functional reactive layer [11]. 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 [12] 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 [13] 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 [14] 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. ARTIS has been experimented in the SIIVIBA [15] and FIPA [16] platforms.

IDEA architecture (Intelligent Distributed Execution Architecture) [17] 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 [14] and IDEA [17] 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 [18] 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. Robot agents can receive estimates of the obstacles from other robots, and commands and specifications from the human operator on input channels, and its can sent its own informations to other robots or to the human operator using the output channels.

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 [19]. 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$^+$ [20], ISLANDER [21], OMNI [22]. 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[4] 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 [23] to plan the robot trajectory, and the Orbital Obstacle Avoidance Algorithm [24] for unexpected obstacles.

Afterwards, the Control Layer controls all the required actuators to follow the computed trajectory until the objective.

3.2 Properties

Communication One of the main objects of study in multi-robot systems research is the communication or interaction between the robots. Three main communication structures are often used [25, 26].

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

- First is communication or interaction via environment: this occurs when the environment itself is the communication medium with no explicit communication between agents. This type of interaction between robots is also known as stigmergy and examples can be found in [27].
- 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, but without explicit communication. An example would be vision by means of omni directional cameras [2].
- 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.

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 OS

As said on section 2, we use Electronic Institution and $\mathcal{M}$OISE^{Inst} [29] as organizational model. $\mathcal{M}$OISE^{Inst} allows to model a MAS with four dimensions : structural, contextual, functional and normative. (cf. Figure **??**)

In other words, thanks to $\mathcal{M}$OISE^{Inst} we can model the fact that :

(FS) to accomplish a specific *goal* or *mission* of the Functional Specification.
(SS) An agent playing a particular *role* of the Structural Specification,
(CS) when the organization is in a given *context* of the Contextual Specification,
(NS) according to a deontic operator of the Normative Specification, it force, allow or forbid actions

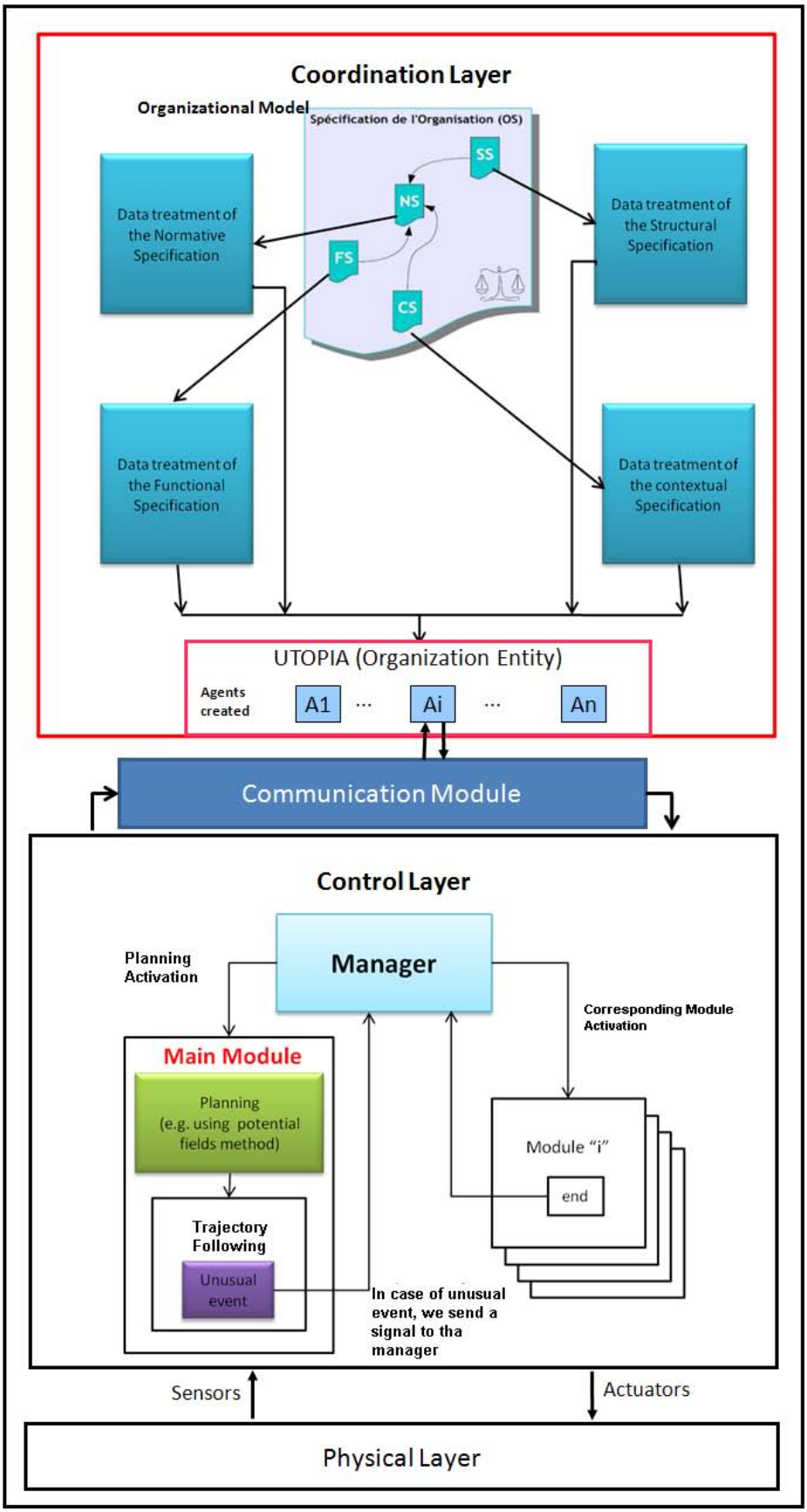

Fig. 1. Architecture of a robot at the individual scope [28]

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 a planning-based mode.

Nevertheless, we also model a possible reactive mode to face unexpected events. The resulting OS allow cooperation and coordination as all the agents are managed by a supervisor detecting and avoiding conflicts. In this particular case, the possible conflicts are collision between robots and the cooperative behavior is the way robots constraint themselves relatively to their own objective to guarantee the accomplishment of the most important social-scope goal : the preservation of all the robots.

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 :

$$mSupervisor = gRoot \parallel \{gSupervisor, gCollisionSolver\}$$

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 $N\{i\}planningMode$ and $N\{i\}reactiveMode$ with $i = \{1..n\}$.

- $N\{i\}planningMode$ 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 $sRobot\{i\}/reactiveMode/Active$ to do the mission *mReact*.

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

Table 1. Normative Specification glueing all three other specifications

5 MAS2CAR's Implementation

In order to be focused on the model and collaborative behavior we use $\mathcal{U}$TOPIA [30] as MAS middleware to implement the Coordination Layer of our architecture.

$\mathcal{U}$TOPIA automatically deploy a MAS that can be specialized with goal or missions corresponding to Java classes run on the fly by the middleware in respect of the Normative Specification.

In other words, the concrete MAS corresponding to the specification seen in section 4, is mainly managed with $\mathcal{U}$TOPIA. We only present the key aspects of the implementation of the goals and information shared between goals and agents.

5.1 Goals implementation

On section 4, we described our OS and we have seen that the Functional Specification implies set of goals called missions. A goal such as *gSupervisor* is abstract and only denote, at this state, a set of actions.

Of course, in our model of implementation, we have associated abstract goal designations with concrete actions. $\mathcal{U}$TOPIA permits to map declared Functional Specification goals with Java classes.

5.2 Information sharing

When a goal is achieved, $\mathcal{U}$TOPIA's agent playing a role of the Structural Specification run the next goal according to the mission.

For a mission such as $mPlan = gReachTarget$
$\rightarrow gAskForPermission$
$\rightarrow gComputeTrajectory$
$\rightarrow gWaitSupervisor$ we can notice that information have to be shared between goals. For instance, after computing a plan in *gComputeTrajectory* and after obtaining supervisor's permission we have to respect the plan in *gReachTarget*. To this end, we obviously need the planning when *gReachTarget* is run, even if the planning isn't computed in this goal.

We use a couple of $\mathcal{U}$TOPIA's primitives to share information between goals run by a same agent. At the end of the goal implementation *gComputeTrajectory* we publish a shared object under the name "planning" describing the plan.

Shared objects are also used by goals running in parallel particularly those of Supervisor. We have an object called *Environment* describing the essential information about robots and obstacles constituing the environment.

Basically, the goal *gSupervisor* update the object according to received messages from robots, *gCollisionSolver* use this global point of view to detect and avoid conflicts by adding constraints to the conflicting planning such as speed constraints, coordinates of a new trajectory...

6 Simulation and results

Figure 2 shows a closer view of the robots represented by filled circles (R1 red, R2 blue and R3 green). In this demonstration, the robots R1 and R2 should have been too close according to our criteria defined by l_{min}. This situation is shown by two unfilled red and green circles in the middle of the picture.

The speed of the R1 robot was reduced in *gComputeTrajectory* according to constraints received by the supervisor's goal *gAvoidCollision*. Unfortunately, an unexpected obstacle was placed right on the robot's trajectory. R1 detected it through its sensors while running *gReachTarget*. That's why you can see this robot in reactive-mode avoiding the obstacle by running *gAvoidObstacle* which is the cycle limite avoidance [24].

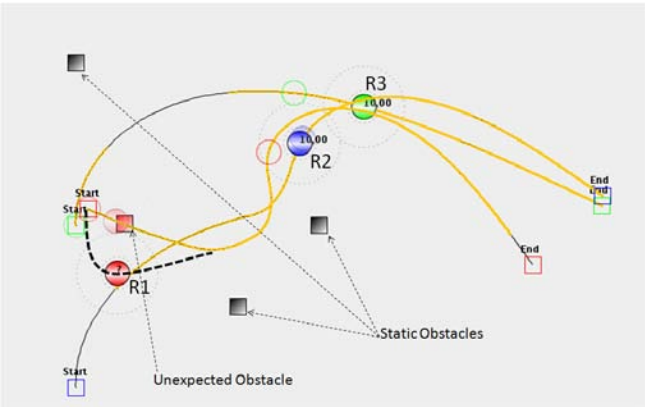

Fig. 2. Reactive-mode in order to avoid an unexpected obstacle

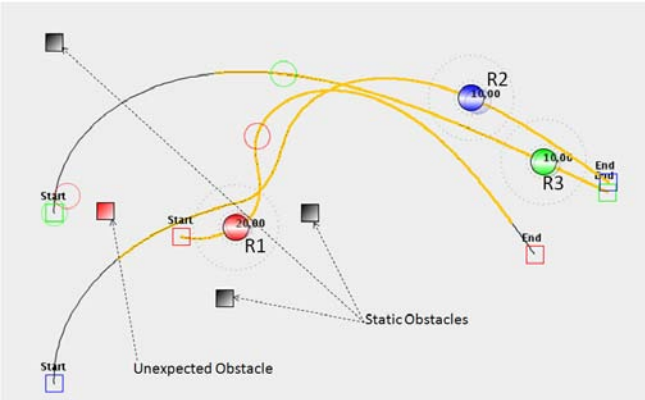

Fig. 3. Robot getting back on its initial trajectory

You can also notice a red transparent circle in collision with the obstacle. This circle represent the expected position of the robot according to the last received planning (this trajectory is obviously not traked by the robot).

In figure 3, R1 avoided successfully the obstacle and is back on its initial trajectory thanks to *gReturnOnTrajectory*. A new trajectory computation is done to reach its goal from its new position, taking into account the position and the speed of the other robots, in this case no conflicts was detected with them (far enough from its position).

7 Conclusion

In this paper we described the proposed architecture model and its three parts. The Physical Layer is related to the robot itself, its sensors, actuators and characteristics. The Control Layer module is in charge of several treatments such as sensors / actuators management and serve as middleware between Physical and Coordination Layers.

We focused on the Coordination Layer and have shown how a multi-agent system can be helpful to bring coordination and cooperation into a multiple heterogeneous robot 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 OS and translate it into a concrete MAS corresponding to the Coordination Layer throught our goal implementations executed by the robots.

Beyound the theory, we shown a concrete and generic OS demonstrating how we can manage the robots in a structured environment while allowing the robots to autonomously react to unexpected events.

Moreover, we presented some of our results showing how the supervisor can send new constraints to avoid conflicts, how transitions are sent to switch between planning and reactive mode while allowing more autonomy for a particular robot, and finally how this robot switch to planning mode again to knuckle back under supervisor.

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 treatment of a big abstract mission... his is the force of using an organizational model such as $\mathcal{M}$OISE^{Inst} which aims, *inter alia*, at structure collaboration.

References

1. Y. U. Cao, A. S. Fukunaga, and A. B. Kahng, "Cooperative mobile robotics: Antecedents and directions," in *Autonomous Robots*, vol. 4, 1997, pp. 1–23.
2. L. Parker, "Alliance: An architecture for fault tolerant multi-robot cooperation," in *IEEE Transactions on Robotics and Automation*, vol. 14, 1998, pp. 220–240.
3. Morrow, J., and Khosla, "Manipulation task primitives for composing robot skills," in *IEEE Int. Conf. on Robotics and Automation*, vol. 14, 1997, pp. 3354–3359.
4. Simmons, R., Singh, S., Hershberger, D., Ramos, J., and Smith, "Coordination of heterogeneous robots for large-scale assembly," in *ISER00, 7th Int. Symposium on Experimental Robotics*, 2000, pp. 311–320.
5. Klavins, E., Koditschek, and D., "A formalism for the composition of concurrent robot behaviors," in *IEEE Int. Conf. on Robotics and Automation*, vol. 4, 2000, pp. 3395–3402.
6. Burridge, R. Rizzi, A., Koditschek, and D., "Sequential composition of dynamically dexterous robot behaviors," in *International Journal of Robotics Research*, vol. 18(6), 1999, pp. 534–555.
7. P. Ridao, M. Carreras, J. Batlle, and J. Ama, "Oca: A new hybrid control architecture for a low cost auv," in *Proceedings of the Control Application in Marine Systems*, 2001.
8. R. Brooks, "A robust layered control system for a mobile robot," in *IEEE Journal of Robotics and Automation*, 1986, pp. 14–23.
9. J. Rosenblatt, "Damn: A distributed architecture for mobile navigation," in *Journal of Experimental and Theoretical Artificial Intelligence*, 1997, pp. 339–360.
10. R. Lumia, J. Fiala, and A. Wavering, "The nasrem robot control system and testbed," in *IEEE Journal of Robotics and Automation*, 1990, pp. 20–26.
11. S. Schneider, V. Chen, G. Pardo-Castellote, and H. Wang, "Controlshell: A software architecture for complex electro-mechanical systems," in *International Journal of Robotics Research, Special issue on Integrated Architectures for Robot Control and Programming*, 1998.
12. A. Farinelli, L. Iocchi, and D. Nardi, "Multirobot systems: a classification focused on coordination," in *Systems, Man and Cybernetics, Part B, IEEE Transactions*, 2004, pp. 2015–2028.
13. H. Bruyninckx, "Open robot control software: The orocos project," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2001)*, vol. 3, 2001, pp. 2523–2528.
14. V. Botti, C. Carrascosa, V. Julian, and J. Soler, "Modelling agents in hard real-time environments," in *Lecture Notes in Computer Science*, vol. 1847/1999, 1999, pp. 83–78.
15. N. R. Jennings, K. Sycara, and M. Wooldridge, "A roadmap of agent research and development," in *Autonomous Agents and Multi-Agents Systems Editorial Kluwer Academic Publishers*, 1998, pp. 7–38.
16. M. Dorigo, V. Trianni, E. Sahin, R. Grob, T. Labella, G. Baldassarre, S. Nolfi, J.L.Debeubourg, F. Mondada, D. Floreano, and L. Gambardella, "Evolving self-organizing behaviors for a swarm-bot," in *Autonomous Robots*, vol. 17, 2004, pp. 223–245.
17. N. Muscettola, G. Dorais, C. Fry, R. Levinson, and C. Plaunt, "Idea: Planning at the core of autonomous reactive agents," in *Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space*, October 2002.

18. R. Fierro and A. Das, "A framework and architecture for multi-robot coordination," in *The international Journal of Robotics Research*, vol. 21, 2002, pp. 977–995.
19. M. Esteva, "Electronic institution: from specification to development," in *IIIA Ph.D. Monography*, vol. 19, 2003.
20. J. F. Hübner, J. S. Sichman, and O. Boissier, "A model for the structural, functional, and deontic specification of organizations in multiagent systems," in *SBIA'02*, ser. LNAI, no. 2507. Springer, 2002, pp. 118–128.
21. M. Esteva, B. Rosell, J. A. Rodriguez-Aguilar, and J. L. Arcos, "Ameli: An agent-based middleware for electronic institutions," in *AAMAS'2004*. New York City, USA: ACM Press, 19-23 July 2004, pp. 236–243.
22. V. Dignum, J. Vazquez-Salceda, and F. Dignum, "Omni: Introducing social structure, norms and ontologies into agent organizations," in *ProMAS International Workshop 2004*, New York, USA, 2004.
23. H. Safadi, "Local path planning using virtual potential field," in *COMP 765, Spatial Representation and Mobile Robotics Project, McGill University, School of Computer Science*, 18 april 2007.
24. L. Adouane, "Orbital obstacle avoidance algorithm for reliable and on-line mobile robot navigation," in *9th Conference on Autonomous Robot Systems and Competitions*, 2009.
25. D. J. Stilwell, B. E. Bishop, and C. A. Sylvester, "Redundant manipulator techniques for partially decentralized path planning and control of a platoon of autonomous vehicles," in *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 35, 2005, pp. 842–848.
26. J. Werfel and R. Nagpal, "Extended stigmergy in collective construction," in *IEEE Intelligent Systems*, vol. 21, 2006, pp. 20–28.
27. A. K. Das, R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer, and C. J. Taylor, "A vision-based formation control framework," in *Robotics and Automation, IEEE Transactions*, vol. 18, 2002, pp. 813–825.
28. M. Mouad, L. Adouane, D. Khadraoui, and P. Martinet, "Multi-agents system to control and coordinate teamworking robots (mas2car)," in *7eme Journees Nationales de la Recherche en Robotique JNRR'09*, France, 2009.
29. B. Gâteau, O. Boissier, D. Khadraoui, and E. Dubois, "Moiseinst: An organizational model for specifying rights and duties of autonomous agents," in *Third European Workshop on Multi-Agent Systems (EUMAS 2005), Brussels, Belgium*, December 7-8 2005.
30. P. Schmitt, C. Bonhomme, and B. Gâteau, "Easy programming of agent based electronic institution with utopia," in *10th international conference on New Technologies of Distributed Systems (NOTERE 2010)*, Tozeur - Tunisia, 31 May 2010.