Multi-agents based system to coordinate mobile teamworking robots

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

Multi-agents based system to coordinate mobile teamworking robots

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

[1] CRP Henri Tudor - SSI dept. Luxembourg
[2] LASMEA - Aubière, France
mehdi.mouad@tudor.lu; lounis.adouane@lasmea.univ-bpclermont.fr

Abstract. This paper aims at presenting the Multi-Agents 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 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. We will present key problematics of the Cooperation Layer implementation solved thanks to $\mathcal{U}$TOPIA and exhibit robotic cooperative behavior related to our scenario through simulation results.

1 Introduction

A multi-robot system can be defined as a set of robots operating in the same work space. In addition, cooperative behavior is settled as: given some task specified by a designer, a multi-robot system displays cooperative behavior if, due to some underlying mechanism (i.e. the mechanism of cooperation or coordination), there is an increase in the total utility of the system.

In mobile robotics, the need to operate in increasingly complex and unstructured environments, and at the same time reduce costs in terms of time or power to a minimum raise the development of autonomous vehicles.

This need for autonomy requires from the vehicle a certain capacity of being able at any moment to assess both its condition and the state of its environment. This informations have to be combined with the different vehicles states and its mission requirements in order to make coherent decisions. That induces a high level of complexity in the robotic control architecture software.

This paper aims at presenting MAS2CAR[3] a new architecture model for multi-robot systems and more particulary its agent-based Coordination Layer.

Robotic software is now one of essential part of robotics system development, therefore software architectures design methods and concepts, often inspired by engineering software field, are necessary within a robotics project. The last few 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 [1], a behavior-based software architecture, has focused on fault tolerant cooperative control.

The coordination of robots for large-scale assembly has been considered in Simmons et al. [2]. Klavins and Koditschek [3] have presented tools for composing hybrid control programs for a class of distributed robotic systems. This approach assumes that a palette of controllers for individual tasks is available. These controllers, i.e, robot behaviors, are sequentially composed using the techniques introduced in Burridge, Rizzi, and Koditschek [4].

Control software architecture design approaches are usually classified into three main categories [5]:

- Reactive 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 input value. [6] [7].
- Hierarchical architectures's design is centered on the decision-making system. These architectures are organized several layers, A layer only communicates directly with the lower layer and the one directly above, this is how the treatment is done[8].
- Hybrid architectures are a mix of the two previous ones [5]. Usually these are structured in three layers: the deliberative layer, based on planning, the control execution layer and a functional reactive layer.

A MAS particularly meet the underlaying needs of supervision and coordination of multi-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[14]. Indeed, this organizational model allow to express cooperation schemes defined by the user with an Organization Modeling Language such as for instance $\mathcal{M}$OISE$^+$ [9], ISLANDER [10], OMNI [11]. 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 *Organization Specification* (OS).

To summarize our different ideas, we state to develop a hybrid architecture which takes into account: the advantages of the other kind of architectures, to obtain more efficient reaction in different aspects, allows coordination and permits hybrid distributed / centralized aspect. This architecture is dedicated to multi-robot systems with a high degree of coordination between autonomous

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

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 constraints and related issues.

This paper is organized as follow : in section 2 we introduce our architecture model in a global point of view and we underline 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 multi-agent systems have been used to achieve this part of the architecture. More particularly, we describe the Organization Specification in section 3, some MAS important implementation aspects in 4 and our simulation results in section 5.

2 MAS2CAR global model

2.1 Properties

Elaborating an innovative architecture to control a group of coordinated autonomous vehicles in unstructured environments, have to consider 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** aims at figuring out the best way to accomplish "basic" goals with modules aiming at plan, re-plan, and manage the sensors and actuators. As shown in Figure 1, the Control Layer makes the link between the Physical Layer and the Coordination Layer
3. The **Coordination Layer** is dedicated to more complex abstract or social goals.

Communication We use interaction via communications [12, 13] in our model 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 Each 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.

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

The Control Layer is in charge of basic decisions 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.

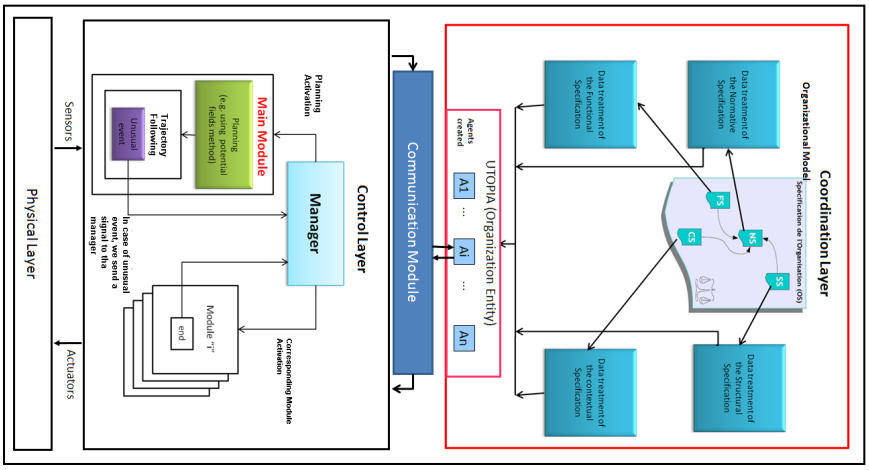

Fig. 1. Architecture of a robot at the individual scope JNRR09[4]

3 MAS2CAR Organization Specification (OS)

In this section we will describe an Organization-oriented approach to model our MAS using $\mathcal{M}$OISE^{Inst} as organizational model. 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.

3.1 Structural Specification

Define a role per robot: $rRobot\{i\}$ with $1 \leqslant i \leqslant n$ and n the total number of robot. One and only one agent is associated to each robots. 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 the MAS, Coordination Layer of our architecture, is mainly composed by the agents representing an associated agent's Control Layer and by a supervisor.

3.2 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*. We have one scene for the supervisor, *sSupervisor*, and one for each robot : $sRobot\{i\}$ (cf. figure 2). Theses scenes includes specific contexts influencing the behavior of the robots: (i)Planning mode associated to the **initial** context *planningMode*, used as often as possible by the robots to compute their trajectories and reach their goals. (ii)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 in $n+1$ different contexts **at the same time**. First, we have: *sSupervisor/life* and *sRobot*$\{i\}$*/planningMode* $\forall i \in [1..n]$. If we send the transition *aO5* ("avoid obstacle" into the scene *sRobot*$\{5\}$) the contexts turn to *sSupervisor/life*, *sRobot5/reactiveMode* and *sRobot*$\{i\}$*/planningMode* $\forall i \in [1..n], i \neq 5$.

3.3 Functional Specification

Here, we have specified *goals* and set of goals (*missions*) for the robots and the supervisor (cf. figure 2). The goals are executed in a specific order according to three modalities : Sequence, Parallelism, and Choice. We have defined three missions :

$mSupervisor = gRoot \parallel \{gSupervisor, gSupervisorGUI, gCollisionSolver\}$: The supervisor have to do three set of actions in parallel : management of the messages and requests from the robots within *gSupervisor*, representation of the robots in a graphical interface within *gSupervisorGUI* and detection / resolution of conflicts between robots within *gCollisionSolver*.

$mPlan = gReachTarget \rightarrow gAskForPermission \rightarrow gComputeTrajectory \rightarrow gWaitSupervisor$: First we execute *gWaitForSupervisor* (is the supervisor here and ready ?) then, we execute *gComputeTrajectory* to 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. Finally, the goal *gReachTarget* is executed and we send messages to the Control Layer in order to move in respect of the planning.

$mReact = gReturnOnTrajectory \rightarrow gAvoidObstacle$: Is used to manage an unexpected obstacle detected by sensors. In *gAvoidObstacle* the avoidance is reactive maintaning a given distance with the obstacle. And run *gReturnOnTrajectory* to join the trajectory planified before and quit the reactive mode.

3.4 Normative Specification

In the normative specification (cf. figure 2), we have a norm for the supervisor *NSupervisor*, which forces the agent playing the role *rSupervisor* when the Institution is in the context *sSupervisor/life* to do the mission *mSupervisor* described before. For the n robots, we have two norms $N\{i\}$*planningMode* and $N\{i\}$*reactiveMode* with $1 \leqslant i \leqslant n$. (1)$N\{i\}$*planningMode* forces the agent playing the role *rRobot*$\{i\}$ when the Institution is in the context *sRobot*$\{i\}$*/planningMode* to do the mission *mPlan*, (2)$N\{i\}$*reactiveMode* forces *rRobot*$\{i\}$ when the Institution is in the context *sRobot*$\{i\}$*/reactiveMode* to do the mission *mReact*.

4 MAS2CAR Implementation

In order to be focused on the model and collaborative behavior we use $\mathcal{U}$TOPIA [14] as MAS middleware to implement the Coordination Layer of our architecture.

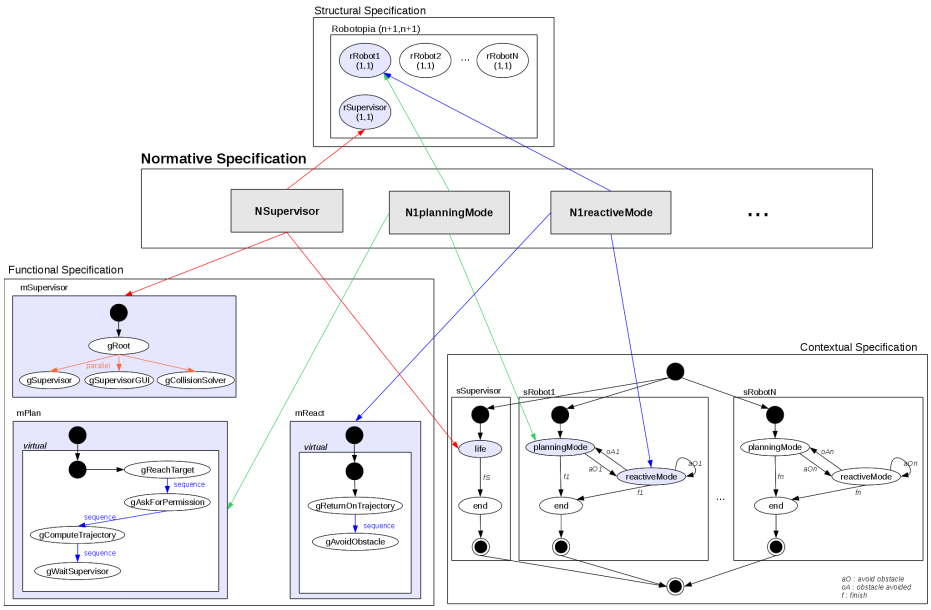

Fig. 2. Normative Specification and links with others specifications

$\mathcal{U}$TOPIA automatically deploy a MAS adapted to a XML Organization Specification. This MAS 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 3, 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.

4.1 Goals implementation

On section 3, we described our Organization Specification and we have seen that the Functional Specification implies goals and set of goals called missions. A goal such as *gSupervisor* is abstract and only denote, at this state, a set of "fuzzy" actions. Of course, in our model implementation, we have associated abstract goal designations with concrete actions.

$\mathcal{U}$TOPIA maps declared FS' goals with Java classes. For instance, *gSupervisor* is mapped this way:

```
<Goal id="gSupervisor" class="robotopia.supervisor.GSupervisor"/>
```

This class itself access $\mathcal{U}$TOPIA's primitives through the implementation of the class "GoalImplementation" overriding a method called "run()".

```
public class GSupervisor extends GoalImplementation{public void run(){ ... }}
```

5 Simulation and results

Figure 3 shows our two modes, the robots are represented by filled circles (red, blue and green). In the planning-mode demonstration, the red and green robots should have been too close. Nevertheless, the speed of the red robot was reduced in *gComputeTrajectory* according to constraints received by the supervisor's goal *gAvoidCollision*, after that they continue there path without collisions. In the reactive part, an unexpected obstacle (represented by a red square) was placed right on the robot's trajectory. Red robot detected it through its sensors while running *gReachTarget*. we can see the switch to reactive-mode, and the avoidance of the obstacle by running *GAvoidObstacle*, and joins back its initial trajectory thanks to *GReturnOnTrajectory*. A new planning was computed to reach its goal from its new position and there was no conflict with this new planning as the two others robot are far enough from its position.

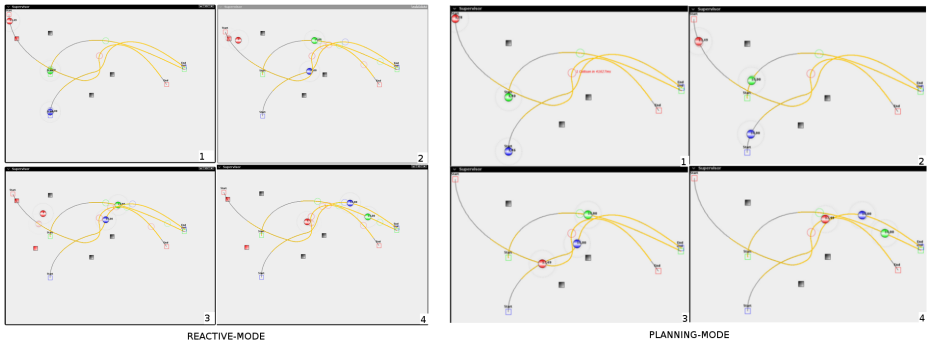

Fig. 3. Reactive-mode and the Planning-mode

6 Conclusion

In this paper we described our architecture model and its three parts. The Physical Layer, the Control Layer and 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 : (i) Using an Organizational Model ($\mathcal{M}$OISE^{Inst}) in order to describe the structure, goals and contexts in a normative Multi-Agent System. (ii) Taking the benefit of $\mathcal{U}$TOPIA framework to handle the OS and translate it into a concrete MAS corresponding to our Coordination Layer throught our goal implementations executed by the robots.

Beyound the theory, we shown a concrete and generic Organization Specification (OS) demonstrating how we can manage the robots in a structured environment while allowing robot autonomous reactions to face unexpected events.

Moreover, we presented some of our results showing how the supervisor can send constraints to avoid conflicts, how transitions are sent to switch between planning and reactive mode allowing 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 as it allows to face every events while keeping a specification of global goals.

References

1. Parker, L.: Alliance: An architecture for fault tolerant multi-robot cooperation. In: IEEE Transactions on Robotics and Automation. Volume 14. (1998) 220–240
2. Simmons, R., Singh, S., Hershberger, D., Ramos, J., Smith: Coordination of heterogeneous robots for large-scale assembly. In: ISER00, 7th Int. Symposium on Experimental Robotics. (2000) 311–320
3. Klavins, E., Koditschek, D.: A formalism for the composition of concurrent robot behaviors. In: IEEE Int. Conf. on Robotics and Automation. Volume 4. (2000) 3395–3402
4. Burridge, Rizzi, R., A., Koditschek, D.: Sequential composition of dynamically dexterous robot behaviors. In: International Journal of Robotics Research. Volume 18(6). (1999) 534–555
5. Ridao, P., Carreras, M., Batlle, J., Ama, J.: Oca: A new hybrid control architecture for a low cost auv. In: Proceedings of the Control Application in Marine Systems. (2001)
6. Brooks, R.: A robust layered control system for a mobile robot. In: IEEE Journal of Robotics and Automation. (1986) 14–23
7. Rosenblatt, J.: Damn: A distributed architecture for mobile navigation. In: Journal of Experimental and Theoretical Artificial Intelligence. (1997) 339–360
8. Lumia, R., Fiala, J., Wavering, A.: The nasrem robot control system and testbed. In: IEEE Journal of Robotics and Automation. (1990) 20–26
9. Hübner, J.F., Sichman, J.S., Boissier, O.: A model for the structural, functional, and deontic specification of organizations in multiagent systems. In: SBIA'02. Number 2507 in LNAI, Springer (2002) 118–128
10. Esteva, M., Rosell, B., Rodriguez-Aguilar, J.A., Arcos, J.L.: Ameli: An agent-based middleware for electronic institutions. In: AAMAS'2004, New York City, USA, ACM Press (19-23 July 2004) 236–243
11. Dignum, V., Vazquez-Salceda, J., Dignum, F.: Omni: Introducing social structure, norms and ontologies into agent organizations. In: ProMAS International Workshop 2004, New York, USA (2004)
12. Stilwell, D.J., Bishop, B.E., Sylvester, C.A.: Redundant manipulator techniques for partially decentralized path planning and control of a platoon of autonomous vehicles. In: IEEE Transactions on Systems, Man, and Cybernetics. Volume 35. (2005) 842–848
13. Werfel, J., Nagpal, R.: Extended stigmergy in collective construction. In: IEEE Intelligent Systems. Volume 21. (2006) 20–28
14. Schmitt, P., Bonhomme, C., Gâteau, B.: 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)