MOBILE ROBOT AGENT ARCHITECTURE BASED ON COMPETITIVE MODULES FOR MULTI ROBOTS APPLICATIONS

Mr. Z. TOUKAL, Pr. Y. AMIRAT, Mr. R. BENAMARA and Dr. A. MELLOUK
LIIA Lab. IUT de Créteil-Vitry, France

99ADM047

Abstract: In multi robots applications, the complexity of a robot's control architecture increases strongly due to its interactions with both the environment and the other robots. We believe that this kind of application has to be considered as a set of processing, communicating and decision-making agents, rather than as a centralized and monolithic organization, where autonomy is underachieved and interactions between the different robots are underemployed. This approach allows a redistribution of the decision-making, leading to a dynamic, reactive and distributed organization. Based on this paradigm, we propose a model of robot control architecture, which exhibits deliberative and reactive features, in order to take into account, on the one hand, the interactions of the robot with the environment, and on the other hand, the impact of these interactions on the decisions of the other robots. This architecture is based on a decomposition in modules of competitive behavior. The interactions of these modules allow to endow the robot with reactive, deliberative and hybrid behaviors. The proposed architecture has been implemented in C++ programming language, using multi-threading techniques under Windows NT. Each module is represented either as a competitive object or as an encapsulation of competitive objects.

1. Introduction

Architectures of control of mobile robots are complex systems. This complexity is due to the necessity to make the mechanisms of knowledge management, and the constraint of reactivity coexist. The diversity of solutions proposed to solve the problem of the control of these robots shows that the different architectures are dedicated to different types of applications. Thus, reactive architectures help to study the emergent behavior of the robot from its primitive behaviors [BRO 91] [MAE 93] [CUE 98] [FER 94]. On the contrary of deliberate architectures, these ones neither allow the planning of operations nor authorize elaborate reasoning [SCH 91][AND 90][STE 93]. This is a major drawback, since in many applications, the robot has to be endowed with capacities of reasoning that confer it the ability to make plans.

In the context of a multi mobile robots application, the control architecture of a robot grows in complexity since this robot interacts with both its environment and the other robots. Such an application has to be considered as a set of decision-making, processing and communications, rather than a centralized and monolithic set, where little room is left to the autonomy and little importance is granted to interactions between the different robots. This agent oriented approach allows a redistribution of the decision-making to the different levels of the organization leading consequently to a distributed, dynamics and reactive organization. On the basis of this paradigm, we propose in this paper concepts that are going to allow a modelization of this organization from both a local point of view (the 'robot' agent component) and a global point of view (interactions between the 'robot' agents). The objective is to propose a generic architecture model of control for each robot, that includes a both deliberative and reactive dimension, in order to consider, on the one hand, its interactions with the environment, and on the other hand, the impact of these interactions on the decisions of the other robots. Given the amount of constraints of a multi - robots application, we propose a control architecture based on the concept of competitive behavior modules [SIM 94][ROS 95][CON 92]. This architecture confers to the robot both deliberative (in order to reason on complex situations) and reactive capacities (in order to respect deadlines). More specifically, it relies on a decomposition in competitive modules which interaction allows to endow the robot of typical reactive, deliberative or hybrid behaviors. This article is structured as following: After a description of the needed characteristics for a mobile robot architecture in a multi-robots context, we present in the second paragraph the concept of competitive behavior modules on which relies the architecture of a 'robot' agent. In the third paragraph, we explain in detail the generic model of the architecture of a 'robot' agent. The implementation of this architecture model for the realization of an exploration task is described in the paragraph 4. It uses the object oriented concepts from C++ language and the multi-threading technique of the Windows NT environment, in order to implement each module as a competitive active object .

2. Concepts of competitive behavior modules

2.1. Characteristics of a mobile robot architecture in a multi - robots application

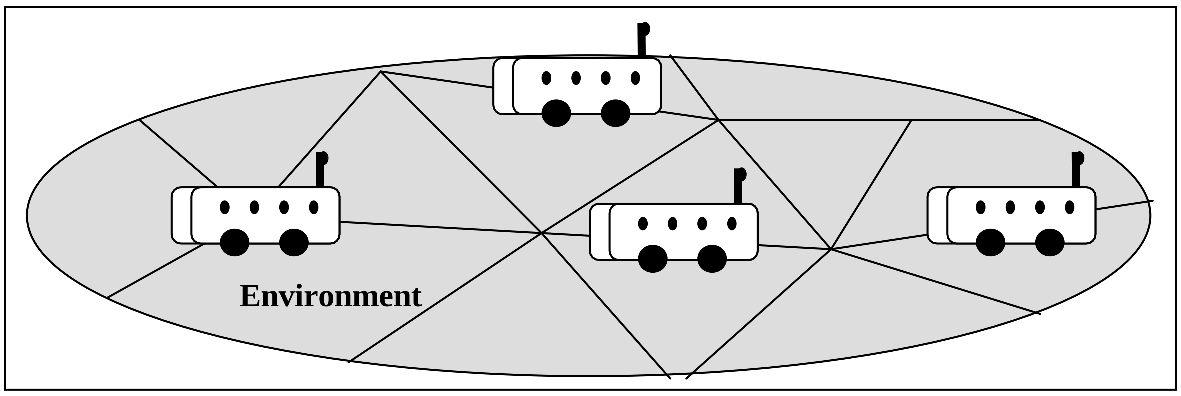

Figure 1. Multi-robot application

The objective of this part is to define a framework in order to achieve the control of a mobile robot organization submitted to heavy constraints: partially known environment, coupling between the operation plans of the robots (figure 1). In such a context, we distinguish two kinds of interaction for each robot: interactions with its environment and with the other robots. In the first case, the environment can be messed up with unknown obstacles which compels to endow each robot with a typical behavior allowing it to reach a targets without entering in collision with an obstacle or another robot. In the second case, we can distinguish several kinds of interaction: the access to spatial resources, the simultaneous perturbation processing and the interaction sharing. An intelligent organization used to explore optimally a partially known environment by sharing known areas is a typical example. The multi-agents concepts, that have been used extensively in research these last years, appear as an interesting and natural approach for the resolution of this type of problems. For this purpose, we have studied the problem of the cooperation between several mobile robots (considered as situated agents) for the management of conflicts that can emerge due to perturbations. The goal was to schedule in real time the disturbed robot's operations, in order to minimize the propagation of the perturbations towards the other robots. We have studied this problem under the constraint of precedence between operations not belonging to the same plan of operations. These works have been concretized by the proposal and the validation of a model of cooperation based on mechanisms of negotiation between robots [TOU 98]. In this paper, we propose concepts that will allow the modelisation of a robot agent in a robots organization. The objective is to propose a generic model of architecture, that integrates both a deliberative and reactive dimension, in order to control each robot, which will allow to consider, in one hand, to take into account its interactions with the environment, and on the other hand, the impact of these interactions on the decisions of the other robots Our objective is to develop a control system that possesses the following properties:

Modularity: The modularity of the control architecture of a mobile robot is achieved by the decomposition in modules that can be developed, implemented, and realized separately. The ability to be reconfigured and to be extended are two characteristics that allow any command system to evolve by the addition of new functionalities and the endowing of a flexibility of adaptation.

Reactivity to the environment: The mobile robot has to be able to manage external asynchronous events in real time so as to respect the dynamics of the environment. An external event can have several origins: presence of an unforeseen obstacle, sudden breakdown, request from an other robot, etc. The reactivity generally implies a real time processing of these events. The real time implies constraints on the reply delays and on some information flows. These constraints depend on the equipment type and the way those events are managed. Thus, the command system has to include the notion of priority and urgency of event processing.

Intelligent behaviors: The intelligence results in perception, reasoning and action capacities The perception translates acquired information into knowledge on the environment. The decisional system generates plans of operations that describe actions to undertake in order to reach objectives of a mission and to react in the face of asynchronous events. The amount of intelligence is closely linked to the different kind of environments in which the robot has to evolve, as well as to the complexity of tasks it has to fulfill. In the context of a multi-robot application where robots have coupled operation plans, the intelligence of the robot can be situated in several levels. The first one is associated to the local environment of the robot.
Thus, in the case of an unknown environment, it is indispensable to endow the robot of an intelligent behavior allowing it to avoid obstacles met on a nominal path. This behavior relies on an on-line control of this path.
The second level of intelligence concerns the sturdiness of the organization, which represents the capacity of the system to manage perturbations malfunctions. More specifically, this level of intelligence confers to the robot some capacities to adapt its behavior in case of perturbation by collaborating with the other robots. As an example, in some situations, a perturbation can entail a delay in the execution of an operation and thereby constrain one or several other robots to be in a situation of exception. Such a contradictory situation can consequently entail the propagation of the perturbation to the other robots, freezing totally the organization. This compels the disturbed robot to have a cooperative behavior, allowing it to negotiate with the other robots a new plan of operations. Such a behavior imposes on the other robots to change their behavior, in order to process the requests of the disturbed robot. Furthermore, each robot has a limited perception of the organization because the knowledge is distributed. Consequently, it is necessary to provide the control system with a mechanism that allows it to manage this distributed and partial knowledge so as to have a coherent global reasoning to produce the most accurate actions.

The third level of intelligence is situated at the control level of the robot's behaviors. It is therefore necessary to have a mechanism that allows changes of strategy in order to adapt the robot's behavior to external events. In other words, this level of intelligence allows to adopt adaptively an adequate behavior of the robot from evaluations of its internal state and those of its environment.

Knowing how difficult it is to realize some tasks in a multi - robots context, we think that it is necessary to appoint a level of reactivity for each level of intelligence. Ways of reasoning have to be introduced in such a way that the robot preserves all its reactivity. Consequently, it is essential to guarantee a competition between behavior modules. In some cases, a situation of conflict between these modules can appear. It is therefore necessary to give the robot a control system that tries to solve these conflicts by authorizing a maximal parallel activity.
In order to fulfill the previously described characteristics, we propose, in what follows, the concept of competitive behavior modules for the definition of a generic architecture model for an 'mobile robot' agent.

2.2. Competitive behaviour modules of a robot agent

Before presenting in detail the elements of the proposed architecture, as well as the mechanisms on which it relies, let's define the concept of behavior module.

2.2.1. Behaviour module

A behavior module is a component that insures a specific function in the control architecture for the achievement of the global goal of an application. It is characterized by its function, its type of granularity, its state, and its interactions with others modules. Moreover, a behavior module can aggregate others behavior modules.

2.2.1.1. Granularity of a behavior module

Depending on the complexity of the module, its activity can take one of the three next forms:

- reactive behaviour, based on skills: It results in a simple reaction to a stimulus by the use of rarely memorized partial or complete information [TIG 96]. The delay of reply is fixed and can be considered as instantaneous. Feedback controlled process (obstacle avoidance, wall following, etc.) or reflex actions (Emergency shutdown, etc.) are a good examples. The module behaves in this case as a limited states automaton without projection of actions in the future .

-deliberative behavior, based on rules: the module relies on the use of rules or memorized procedures also called decisional looped process [GIR 93], which depend on the context. In this case, the module uses data that represent the current situation of the agent (complete sensorial space). It can only be triggered if the situation shows an unknown aspect. In this case, the delay of reply should be bounded.

-deliberative behavior, based on knowledge: It characterizes the behavior in a new situation. The latter requires the elaboration or the modification of an operation plan, depending on the targeted goal, and a mental model of the controlled system. This model allows to evaluate effects of the elaborated plan on the robot itself, and on its environment (the other robots). Furthermore, the information is global and memorized under the shape of a representation of the environment (modelisation phase). The delay of reply should, once again, be bounded.

Depending on its granularity, a module can have either of these two behaviors : procedural (stimuli-answers), or knowledge based. The knowledge base can contain rules, heuristics, specific declarative or procedural knowledge, specific to the knowledge domain of the module. At last, in the case of a rule based system, the induction can be based, depending on the kind of behavior, on an engine of inference of order 0+, order 1, fuzzy, etc.

2.2.1.2. State of a behaviour module.

A module is characterized by its state (active, inactive, in wait of an event), which insures in a way the competition between modules. A module is said to be active if it is processing data, inactive if it is not created yet, and awaiting an event if it is in stand by. When it is active, a module can treat its messages and requests in several ways:
-Immediate processing : At the arrival of an urgent message or event coming from another module, the addressed module activates instantly the memorized procedures associated to this message or event.
- Differed processing: When a normal message or event occurs, the associated treatment can be delayed as specified by a priority based mechanism.
A communication server, associated to the communication module is a typical example of a situation where these mechanisms can be implemented. Such a server should be able to consider several simultaneous requests, and thus, some kind of reentrance.
The competition grants each module a certain amount of autonomy, because it doesn't need external resources to be activated. This autonomy allows the module to control the treatment of the received messages.

2.2.1.3. Interactions between behaviour modules.

A behavior module has an interface which allows it to communicate with other modules, in order to accomplish its mission. Two kinds of communications can be distinguished in the interaction between two modules : The first one consists in a message exchange through a mailbox. The second one relies on a knowledge sharing in a centralized blackboard. This sharing can be set either with or without the involvement of the agent's activities' supervisor.

2.2. 3. Aggregation of behavior modules.

In some situations, a behavior module can need the expertise of several other behavior modules. This results in the activation of one or several behaviors, which will be associated to realize one activity. This activation mechanism is seen as a service begged by a module. The call to the services of the other modules has in and out parameters on the side of the calling module. An operation sequencing module for 'robot' agent is a typical example of behavior encapsulation. Such a module uses the services of reasoning and communication modules. The navigation module, which aggregates the perception, reasoning, and action behaviors, is another example. Control rules are necessary to specify the mechanisms of the communication between these modules.

2.2. 4. The meta - behaviour

Among all the behavior modules, a special module is defined, which is called meta-behavior or supervision module (Figure 2). This module must specify the behavior of the 'robot' agent according to its internal state, and according to its surrounding. This specification too results from the invocation of the reasoning module, using the meta-rules.

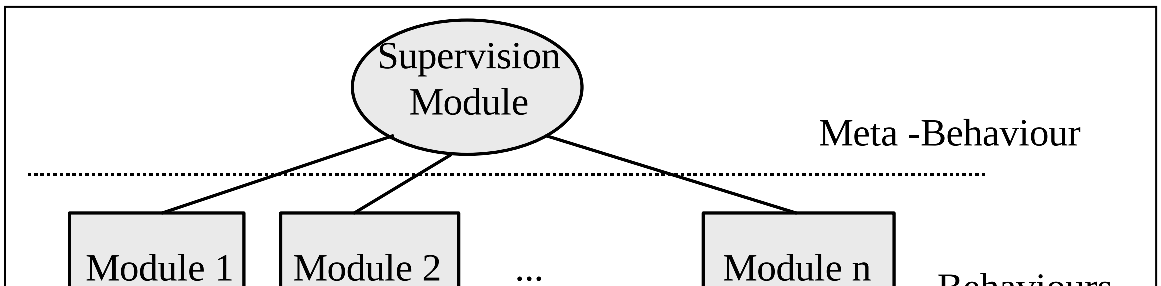

Figure 2. Behaviours of an agent robot

3. Generic architecture of a 'mobile robot' agent

For the synthesis and the implementation of a 'robot' agent, our approach relies on a splitting method of the architecture into competitive behavior modules, and on the combination of objects into these modules. The proposed architecture allows three kinds of reactivity : reflex reactivity, tactical reactivity, and strategic reactivity (Figure 3). It is structured around the following essential modules :

– **Perception module :** It allows to collect the data coming from the various sensors that equip the mobile robot (Odometer, infrared, ultrasound sensors), and to send it to the tactical behavior module.

– **Communication module :** This between agents communication module is charged to elaborate the data frames to be sent to the agent's supervisor, and the checking of the integrity of the received frames. It uses a client-server type communication model.

– **Action module :** It ensures the execution of the mobile robot's displacement orders. Its task is to realize the slaving of the robot's actuators.

– **Reasoning module :** It represents the decisional center of the various abilities of the agent. According to the triggered behavior module, this module determines a convenient response to the 'robot' agent's current situation. The agent's response depends on the know-how and the granularity type of the behavior module. The reasoning module is typically invoked during the negotiation that follows a request transmitted by a disturbed robot, via the communication module. The navigation, where a 'robot' agent has to determine a path without collision to go from one place to another is another example.

– **Security module :** This module allows do give the 'robot' agent a stimuli-response like reflex reactive behavior. This natural behavior triggers an emergency shutdown of the 'robot' agent when the latter is next to an unforeseen obstacle. This leads to a very short delay, as far as the reasoning is reduced to a reflex action. Such a property is obtained thanks to a strong union between the actuators and the sensors through a simple transfer function.

– **Navigation module :** It rules the behavior of the 'robot' agent in the accomplishment of elementary missions (point to point move). It is endowed with the ability to react skillfully to unknown surrounding.. Each mission consists in the computation of a collisionless path by invoking the reasoning module. This module implements a tactical reactivity, because it allows to elaborate online a path without collision by computing on a knowledge base, from the sensors information.

- **Negotiation module (strategic behaviour) :** This strategic behavior module is invoked by the scheduling module in case of disturbance. It is charged to determine the priority of the operations within the disturbed robot. The priority is set after a deliberation on a knowledge base coming from the information extracted from the other ‘robot’ agents. Its way of working is equivalent to a distributed blackboard where each knowledge source is represented by a ‘robot’ agent, the blackboard’s control module being on the same level as the disturbed robot. The activation and control of its knowledge sources is proceeded by either dynamic or static priority.

- **Supervisor module :** This module defines the meta - behavior of the ‘robot’ agent. It manages the data flow, as much as the perturbations. It also manages the asynchronous events coming from the environment. Moreover, it allows to adapt an appropriate behavior by aggregating several behavior modules in front of special situations. It also manages the activation of a behavior module, and transfers the control to it.

- **Sequencing module :** It rules the perturbations resulting from a real-time scheduling. This module uses two solutions : The first one consists in the setting of an heuristic based scheduling, using the local information available at the level of the disturbed ‘robot’ agent. The second passes through the implementation of a negotiation mechanism with the other ‘robot’ agents, in order to minimize the propagation of the perturbation to these agents. In that case, this module sets up a new scheduling and calls the negotiation module.

This architecture presents obvious advantages compared to other approaches. These advantages can be summarized as follows :

– The existence of a strategic behavior, which appears during the negotiation phases. Such a behavior allows to extract a strong global behavior of the organization. This behavior uses a double vision : local (‘robot’ agent), and global (the other ‘robot’ agents).

– The existence of a two level cooperation model : tactical, and strategic cooperation. It consists in giving the robot a management module for its interactions with the other robots. In this case, in the command loop, the perception isn’t connected to local information any more (sensors measures), but to message exchanges with the other robots.

– The introduction of three levels of reactivity : reflex, tactic, and strategic. A type of granularity is associated to each level. The strategic level allows to extract a global behavior from control coordinated at the level of the disturbed ‘robot’ agent. Indeed, each one of the other ‘robot’ agents can be considered as a knowledge source activated by the happening of a perturbation type event. The computation of a new plan relies on the use of a blackboard at the level of the disturbed ‘robot’ agent, and on the use of a strategic cooperation.

– The communication can be established along two orthogonal axis : The first one is horizontal, and associated to the communications between the ‘robot’ agents. These communications are guided by the structure of the organization of the ‘robot’ agents’ society. The structure is responsible for the scheme of interaction of the system. The second axis is vertical and associated to the communications between the modules of a same ‘robot’ agent. This axis is essential to the performance of a system distributed on several levels of abstraction, in order to be able to split the events on the entrance (mechanism of diffusion of a single piece of information to a set of agents), and, on the contrary, to be able to merge atomic level behaviors into high level behaviors.

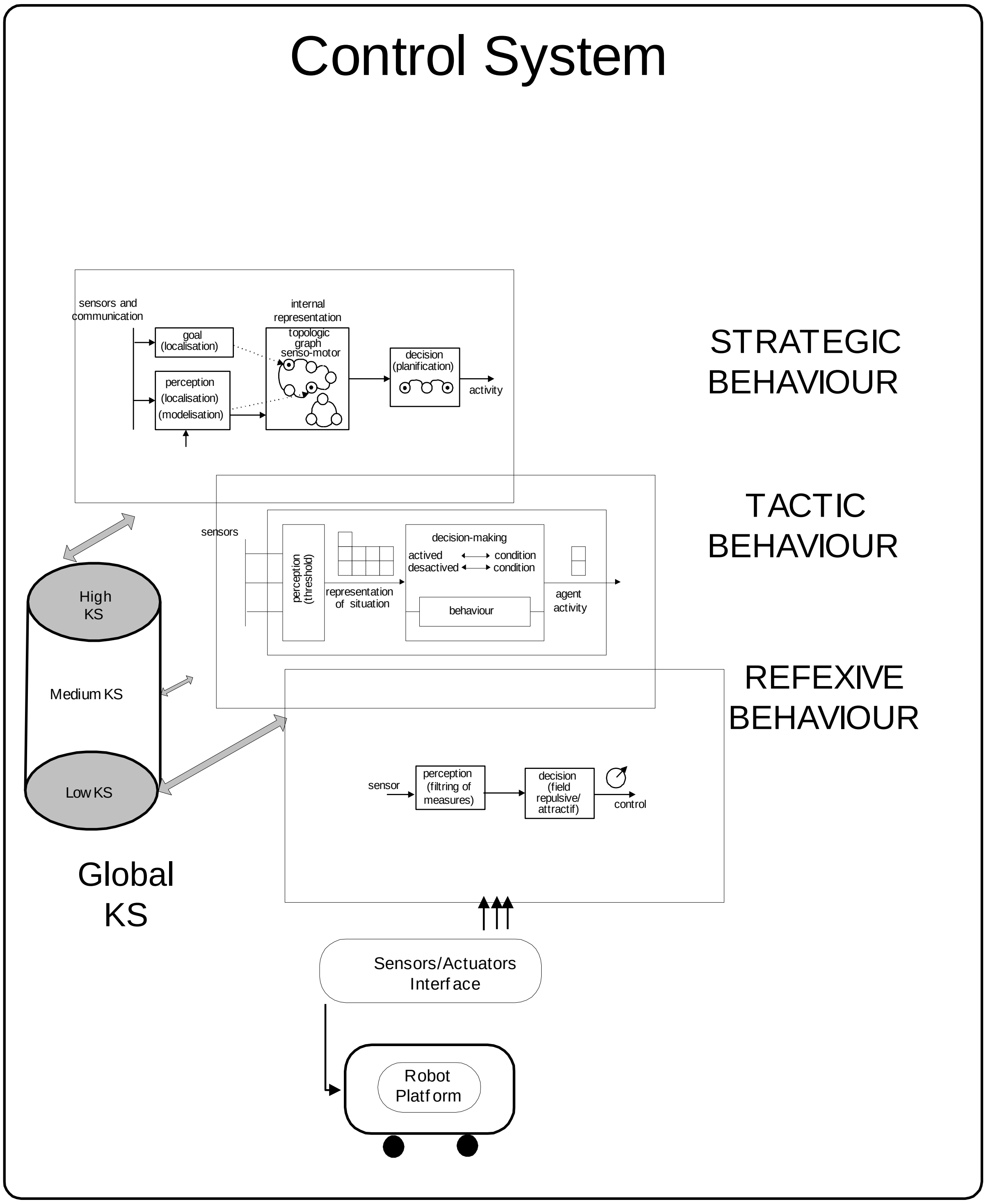

Figure 3 : Control architecture of an agent robot

4. Implementation

4.1. Support of implementation

To build our implementation model, we have chosen an object oriented solution, for two main reasons :

- The object oriented approach offers some great advantages for the realization of a control system based on the behaviors. It specifically allows to encapsulate of inherit behaviors from the remaining system. The software's conceiver doesn't have to know the details of each behavior module and can easily modify and improve them. Moreover, a new module can be easily built by using the principle of inheritance of an existing module. The maintenance of these modules is therefore simplified by this concept.

- The second reason is linked to the implementation of multi-agent systems, for which the majority of the conceptual an theoretic models have been implemented thanks to competitive objects. The competitive object can be considered as a basic element for the implementation of competitive modules. In our approach, a module is naturally implemented as an object, or an encapsulation of competitive objects. Each module has a limited autonomy, as far as its activities are not reduced to a single reception/emission of messages. Actually, a behavior module dedicated to the control of the 'robot' agent must be able to control all these activities (perception, reasoning, communication, ...) in reaction to the requests of the messages.

If the modularity and the advantages of the object oriented concept are interesting for the conception and the reuse of the software, the concurrence and the reactivity necessary to the different behaviors need the implementation of a few mechanisms. It becomes then necessary to define the treatment units that can be executed autonomously and simultaneously (a process, a task, a thread, ...), for the following reasons :

- The application is split into several competitive modules that can be compared to treatment units.
- The different modules must be able to communicate together.
- The modules must respond to asynchronous external events.
- Some requests sent by a module must be treated in a delay compatible with the robot's dynamic.
- Some units of treatment must be able to act on other units (creation, suspension, wake-up, ...), in order to optimize the system resources.
- The treatment units may need a scheduling, with a priority criterion.

The splitting into parallel treatment units doesn't usually have a correlation with the object oriented model, and imposes a specific distribution of the classes between the treatment units. In general, each treatment unit is represented by a class, of which the instance represents the execution (for an example, a process, a task, or a thread). Another way of proceeding consists in associating an execution unit to each instance of an object.

The former criteria (object orientation and competition) have led us to choose a PC as hardware basis, and the use of a multithread operating system (Windows NT), on which the object layer is implemented in C++. Actually, Windows NT grants some facilities to manage the multi-threading, its kernel contains the mechanisms that will allow to manage the concurrence, the synchronization, and the communication between the modules of the system. The Windows NT kernel is responsible of the scheduling of the threads (interruption or preemption of the threads of higher priority), of the synchronization of the process, and the management of the hardware exceptions. Let's notice that some of the kernel's objects are exported at the user-level (API). There are two types of objects : dispatch objects, and control objects. Dispatch objects control the dispatching and the synchronization of the system operations (timers, events, mutual exclusion, semaphore, and threads). The control objects control the kernel's operations without affecting the dispatch objects (Asynchronous Procedure Call, interruptions, process, and profiles)

The 'robot' agent model developed in the following paragraph uses the mechanisms of multi-threading provided by the operating system Windows NT. These tasks implement the architecture's modules and are, for the most part, implemented in a Windows NT thread. This allows to use the associated real time mechanisms, allowing to allocate them priority levels taken in charge by the operating system, to interrupt them, or to synchronize them with signals. The communications between modules will be asynchronous, messages will be used. In this implementation, the 'robot' agents will be distributed on discrete stations.

4.2. Class model

In our approach, the modules are implemented with objects that are considered as autonomous active elements. The notion of task, process, or thread, which intervenes to implement the concept of the concurrence of the treatments, does not result from an object oriented model. All the objects (concrete or abstract) that take part in the architecture of a 'robot' agent are instances of C++ classes. The hierarchic organization of the main classes is represented Figure 4.

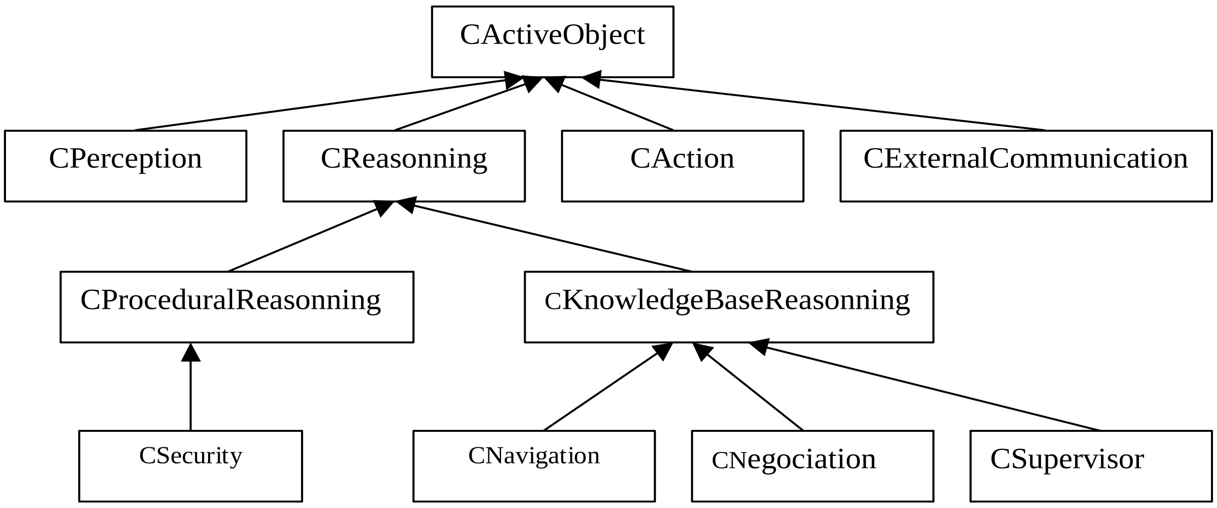

Figure 4. Class hierarchy of the architecture of a 'robot' agent

Each instance of the CActiveObject class (active objects) has a communication interface under the shape of the instance of a CInternalCommunication class. This class allows to implement a client-server type mechanism of communication between active objects. The CActiveObject class describes the behavior and the answer to the requests he is transmitted to by another active object.
In our application, a 'robot' agent is an instance of the CAgentRobot. Each instance of CAgentRobot is composed of :

- An object (instance of the CNavigation sub-class of CKnowledgeBaseReasoning), which describes the behavior of the agent during the navigation phase.
- An object (instance of the CNegotiation sub-class of CKnowledgeBaseReasoning), which describes the behavior of the agent during the negotiation phase.
- An object (instance of the CSecurity sub-class of CProceduralReasoning), which describes the behavior of the agent during the reflex activity phase.
- An object (instance of the CSupervisor sub-class of CKnowledgeBaseReasoning), which describes the meta-behavior of the 'robot' agent.
- Objects that describe the basic behavior modules of the 'robot' agent (Perception, Action, ExternalCommunication).

Concerning the reasoning, the most generic classes are :

- CProceduralReasoning, which describes the procedural modules. The security module uses a signal treatment to watch for the presence of obstacles next to the robot and provoke a reflex stop.
- CKnowledgeBaseReasoning, which implements the knowledge-based modules. The navigation module uses the CKnowledgeBaseReasoning class in order to infer on the base of fuzzy rules for the control of the robot's navigation. In the same way, the Negotiation module uses the CKnowledgeBaseReasoning class to reschedule the operations, starting with a heuristic-based base of knowledge.

4.3. Multi-robot platform simulator

To implement and validate the previously described models, a simulator has been developed. Thanks to the simulation environment written in Microsoft Visual C++, it is possible to parameter the missions of each robot, to visualize and study the behavior of the robots. The environment of simulation is composed with the following modules :

- Object storage module : obstacle, robots, characteristics, and associated data.
- Data input and modification module.
- Sensors and robot command simulation module.
- Visualization module.
- Archiving of each robot's mission module.

The application that has been realized consists in an exploration of an area by several robots. The plans of operations are given to the robots at the startup of the application by a leader robot, and may show intercouplings. Actually, precedence constraints can may appear between operations assigned to robots, and consequently, any delay with the execution of an operation can provoke a conflict with another robot. The evolution environment of each robot is unknown at the beginning of the mission. In order to simulate the happening of a perturbation during the progression of a robot's operation plan, we have simulated the apparition of obstacles on the robot's path. The robot's behavior, which is necessarily avoidance type will generate a delay in the operation plan, and therefore, a conflict. In order to validate the negotiation behavior module, we have implemented various scenarios, representing the different states of a 'robot' agent, which allowed us to extract an appearance phenomenon.

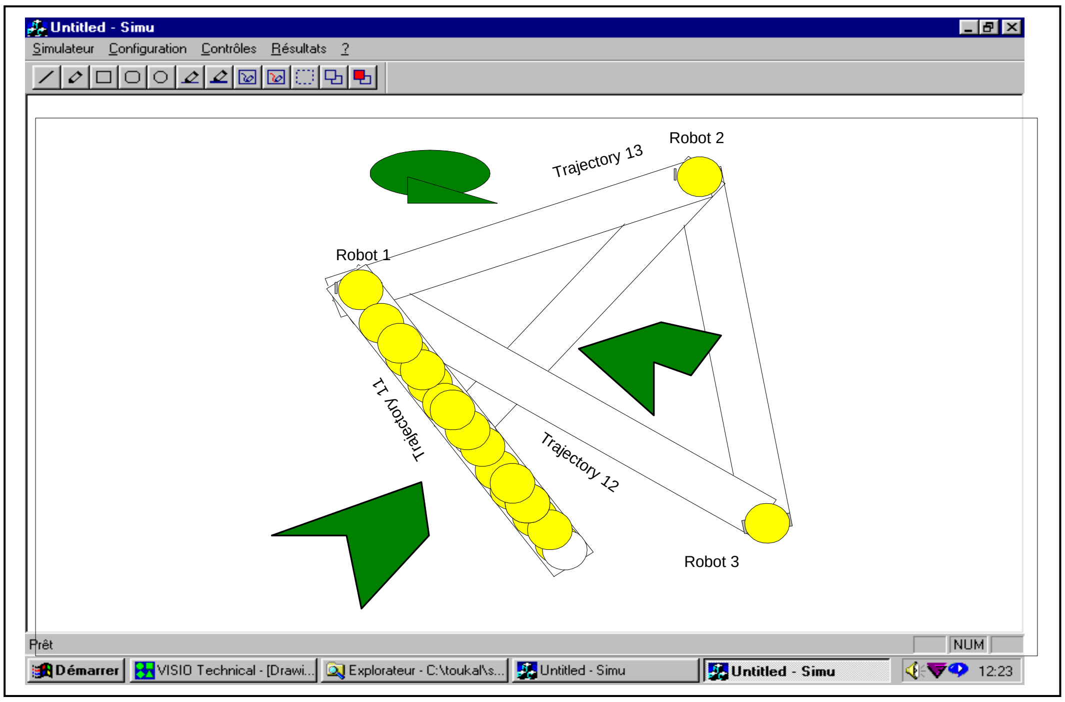

Figure5. Screen of simulation: robot, obstacle, and trajectories

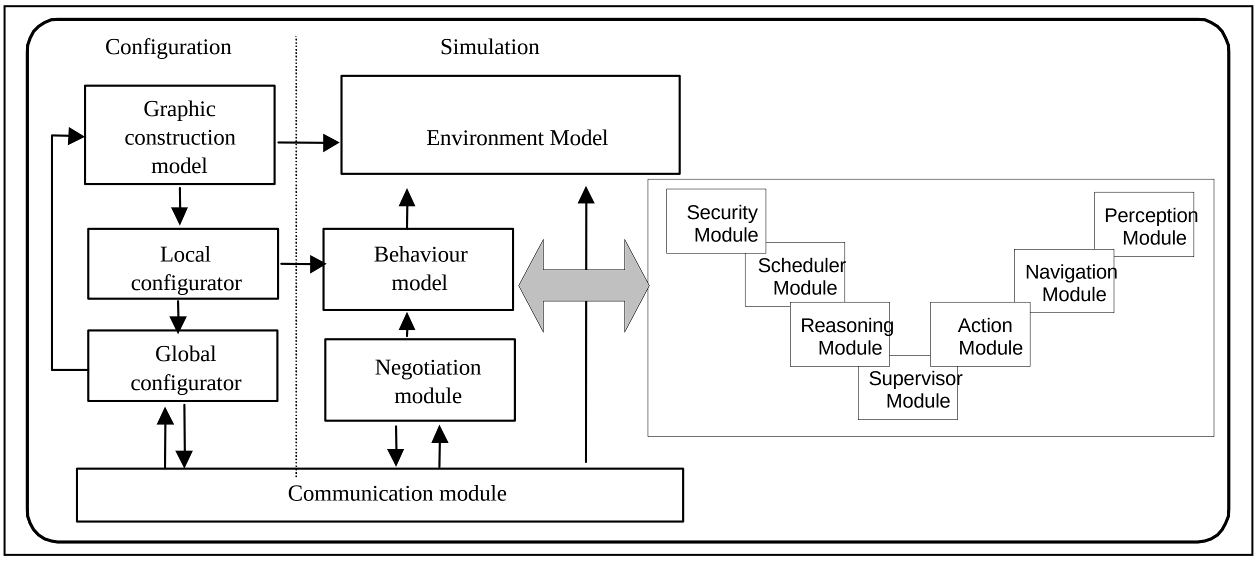

Figure 6. Simulator synopsis

5. Conclusion

We have proposed in this paper a generic model of architecture for a ‘robot’ agent, in the context of a multi-robot application, implying a strong interaction between agents. This module based architecture of behavior shows three levels of reactivity : Reflex, tactic, and strategic. The strategic level, which appears during the phase of negotiations ‘robot’ agents, allows the appearance of a global strong behavior of the organization. This behavior relies on two visions : local (‘robot’ agent), and global (the other ‘robot’ agents). For the implementation of this architecture, each module of behavior is set as a competitive active object. This model of architecture has been used for the realization of the exploration of an area by several robots. In order to validate the behavior modules of the robot, and particularly the negotiation one, we have implemented various scenarios representing the different states of a ‘robot’ agent, which has brought us to extract an emergence phenomenon. Among the interesting applications of the work, we can quote the treatment of new constraints by multi-criteria merging in the modules of negotiation and scheduling. Another one is the introduction of a learning mechanism for the appearance of specific behavior of the organisation.

References

[AND 90] T. L. ANDERSON and M. DONATH. Animal behavior as a paradigm for developing robot autonomy. Robotics and autonomous systems, 6(1-2): 145-168, June 1990.
[BEN 96] R. BENAMARA, C. MORENO. Control agents. Implementing on industrials networks. Expersys' 97, Expert system application and artificial intelligence, IIIT conference, pp. 219-227, Champs/Marne, France, October 1996.
[BRO 91] R.A. BROOKS, Intelligence without representation. Artificial intelligence, vol. 47, 1991, pp. 139-159.
[CON 92] J.H. CONNEL, SSS: A hybrid architecture applied to robot navigation, IEEE Conference on robotics and automation, PP. 2719-2724, 1992.
[CUE 98] J. CUERVO, Une architecture de contrôle et de commande pour la conduite d'un robot mobile autonome: architecture de commande orientée comportements. Thèse de doctorat de l'université d'Evry Val d'Essonne France, 1998.
[FER 94] J. FERBER Simulating with reactive agents. In: E. Hillebrand, J. Stender (Eds) Many Agent Simulation and Artificial lfe, 8-28. IOS Press, Amsterdam, 1994.
[GIR 93] G. GIRALT. Robots d'intervention: Autonomie, réactivité, Programmation. Technical Report 93151, Laboratoire d'automatique et d'analyse des systèmes. Toulouse, April 1993.

[MAE 91] P. MAES, How to do the right thing? Thesis of Ph.D., MIT, 1991.
[ROS 95] J. K. ROSENBLATT and C. THORPE, Combining multiple goals in a behavior-based architecture, proc. Of 1995 International Conference on intelligent robots and systems, Pittsburgh, PA, August 1995, pp. 136-141.
[SCH 91] U. SCHNEPF, Robot Ethology: A proposal for the research into intelligent autonomous systems. In J. A. Meyer and S. W. Wilson, editors, Proceedings of the first international conference on simulation of adaptative behavior: from animals to animates, pages 465-474, 1991.
[SIM 94] R.G. SIMMONS. Structured control for autonomous robots. IEEE transactions on robotics and automations, vol. 10, N°. 1, February 1994.
[STE 93] L. Steels. Building agents with autonomous behavior systems. The artificial life route to intelligence, 1993.
[TOU 98] Z. TOUKAL, Y. AMIRAT, A distributed architecture model to for the driving and the control of a robotized manufacturing system, Proceedings of the 31th International Symposium On Automotive Technology and Automation, Dusserldorf, Allemagne, 2nd-5th June 1998.
[TIG 96] J.Y. TIGLI, Vers une architecture de contrôle pour Robot Mobile orientée Comportement, SMACH, Thèse de doctorat de l'université de NICE-SOPHIA ANTIPOLIS, France, 1996.