# DESIGNING AN INTELLIGENT CONTROL ARCHITECTURE FOR AUTONOMOUS ROBOTS

R.ALAMI, R. CHATILA

LAAS REPORT 93456

NOVEMBER 1993

---

# Designing an Intelligent Control Architecture for Autonomous Robots

Rachid Alami and Raja Chatila
LAAS-CNRS
7, Ave. du Colonel Roche, 31077 Toulouse cedex - France

Bernard Espiau
INRIA
2004, Avenue des Lucioles, 06902 Sophia-Antipolis - France

**Abstract:** Designing a control architecture for autonomous robots able to reason and to act in their environment to accomplish tasks, and to adapt in real time to the actual execution context, raises major questions in the field of control theory, knowledge-based and reasoning systems, as well as software architecture. We propose in this paper an approach to build such systems, based on the experiences and contributions of two laboratories, INRIA and LAAS-CNRS.

## 1 Introduction

It is known that the necessity of designing truly autonomous robots comes from highly demanding applications (undersea intervention, planetary exploration, working in nuclear plants, etc.). Since, in such application fields, any repairing intervention is excluded, the need for a total *technological* autonomy is obvious: energy, computing facilities, sensors... Furthermore, the dialogue with an end-user is reduced to mission specification and, when possible, to very few high-level communications (for example teleprogramming). This means that an almost total *operational* autonomy is also required, especially in the domains of perception, decision and control.

This search for operational autonomy raises major questions in the fields of control theory, knowledge-based and reasoning systems, as well as architecture. However, the specificity of the application domain generates particular constraints which may sometimes be felt as antagonistic according to the concerned scientific domain:

1) The need for *adaptivity* in time and space. This leads to the necessity of on-line planning:

- because of the incompleteness of the knowledge, or of existing uncertainties on measurements. This is for example the case when the environment is gradually discovered by the robot;
- because of technical limitations. In most cases, the on-board storage capabilities make impossible to keep in memory all possible execution courses.

2) The need for *reliability*. Since no intervention is possible in general, it is necessary to be sure, as far as possible, that the robot will work satisfactorily *before its launching*. This implies:

- the necessity of ensuring that every step down to the real-time implementation of the tasks handled by a planner is correct, takes the expected time and that possible conflicts may be avoided;
- to be able to verify the largest possible set of assertions about the mission: nominal behavior, emergency procedures, etc.

Up to now, and in most of the architectures for autonomous robots, these requirements were not considered as compatible: item 2 needs an exhaustive knowledge of all the handled entities and of the way they are organized. Determinism and synchrony assumption are often used. Analysis of continuous-time aspects is also required. This may be clearly not compatible with issues of item 1. Conversely, item 1 requirements make difficult non-trivial verifications and are inappropriate to take into account automatic control aspects.

The aim of this paper is to present and illustrate some ideas which would constitute the generic basis of an architecture allowing to take into account as far as possible the previous requirements. The challenge may here be understood as the feasibility of designing an autonomous robot endowed with capacities of planning its own actions in order to accomplish specified tasks while having a so-called reactive or reflex behavior with respect to its environment. For that purpose, we emphasize design aspects at two levels in the robot architecture:

- the *functional* level: it concerns the design, validation and implementation of control loops associated with a local behavior, called robot-tasks;
- the *decisional level*: it relies on a layered plan-based architecture and provides with a suitable framework for the interaction between *deliberation* and *action*, as defined later.

The interest of this structure is that on-line planning will be allowed at the last level, without excluding reactivity or sensor-based reflex actions at the first one. Furthermore, verification aspects will be made possible almost from implementation up to high

level mission specification when models based on state-transitions systems are used.

The paper is organized as follows: in the next section, we present the concepts underlying our approach to the design of a intelligent robot. Then, in section 3, we detail the functional level, and in section 4 the decisional level. An example of the proposed architecture is finally given in section 5.

## 2 Basic Principles

The proposed approach is based on a few simple ideas, an overview of which we give in this section.

### 2.1 At the Functional Level

A key question is "What is an action?" In other words, what are the characteristics of the smallest entity of this type handled by the decisional level? A partial response lies in the concept of *reactivity*, widely used, sometimes wrongly. The theory of discrete-event systems provide us with a rather precise definition: a synchronous reactive system produces a set of output events deterministically and instantaneously upon the occurrence of a set of input events. Such an emission of signals may thus be considered as an *internal* action, which will perhaps lead to a *robot* action, i.e. a dedicated motion. This last may therefore be started by an event occurrence, but it still remains to describe what this motion will be. A second point is that, clearly, other types of actions may also be triggered on event reception at a higher level: sensor activation, planning requirement, asking for operator intervention, etc. We thus already conclude: 1- that a too general concept of action is not the adequate one at the functional level; 2- that the specification in terms of reactivity only is necessary but not sufficient.

Let us now come to the problem of specifying and controlling a robot action. This may be stated as a control problem which may be efficiently solved in real-time using adequate feedback control loops. Since it is a powerful and mathematically rigorous tool, we believe that control theory should be used as far as possible. In particular, sensor-based control loops are a nice way of performing motions in continuous interaction with the environment, called *reflex actions*. Furthermore, it will be seen that implementation issues are easily handled within this framework. This is why we define the key entity at the functional level as a kind of *event-driven reflex action*, called *robot-task*.

### 2.2 At the Decisional Level

The investigation on the interaction between deliberation and action is certainly a key aspect in the development of intelligent agents and particularly autonomous robots. Deliberation here is both a goal-oriented planning process wherein the robot anticipates its actions and the evolution of the world, and also a time-bounded context dependent decision for a timely response to events.

While there are high emergency situations where a first and immediate (reflex) reaction should be performed, such situations often require last resort safety actions. They can often be avoided if the agent is able to detect events which allow it to predict such situations in advance. Note that this is first a requirement for sensors and sensor data interpretation. Being informed in advance and consequently having more time to deliberate, the agent should be able to produce a better decision. Note also that such a capacity is generally ignored in "purely" reactive approaches where the robot makes uses only of "short" range sensors giving it data on its immediate environment.

Acting is permanent, and planning should be done in parallel: an intelligent agent should not neglect any opportunity to anticipate (i.e, to plan). However, since planning requires an amount of time usually longer than the dynamics imposed by the occurrence of an event, the paradigm that we shall develop consists in controlling the functional level by a deliberative system that has a bounded reaction time for a first response. This level is composed of a *planner* which produces the sequence of actions necessary to achieve a given task or to reach a given goal, and a *supervisor* which actually interacts with the functional level, controls the execution of the plan and reacts to incoming events.

The decisional level may not be unique: a planner usually requires abstract models, and its representations of actions do not embed the *actual* interactions with the environment. For example, a planner would use a model in which situations are described in terms of predicates and general topology (e.g. "connects (D1, R1, R2)") without taking into account the geometry of the environment. Furthermore, there may be several ways to execute a given action (as defined at the high-level planning system) depending on the actual execution situation. Hence, there may be several decisional layers the lower ones manipulating representations of actions which are more procedural and closer to the execution conditions.

Let us now present functional and decisional levels more deeply.

## 3 The Functional Level

As previously evoked, this level mainly relies on the concept of *robot task*. This keystone concept is the minimal granule to be handled by the *decisional* level, while it is the object of maximum complexity to be concerned by the *control* aspects. It characterizes in a structured way a closed loop control scheme, the temporal features related to its implementation and the management of associated events. Fully described in [19], it is defined in a formal way as follows:

*A Robot-Task* is the entire parametrized specification:
- of *an elementary servo-control task*, i.e. the activation of a control scheme structurally invariant along the task duration;
- and of *a logical behavior* associated with a set of signals liable to occur previously to and during the task execution.

Let us give some details on these two aspects.

### 3.1 Design and Implementation of a Servoing Task

A first step consists in specifying the continuous-time control law associated with a dedicated robot-task. In the case of a two-wheeled mobile robot, it is for example the expression of the control which, given desired and actual configuration at time $t$, computes the wheel velocities or the driving torques to be applied. Usually, such a scheme may be split in several functional modules (position estimation, trajectory generation, feedback control...) which exchange data.

Now this description should take into account implementation issues: discretization, variable quantization, delays, computation times, periods, communication and synchronization between the involved processes. This is done by defining the basic entity called *Module-Task*, which is a real-time task used to implement an elementary functional module of the control law.

Since the *Module-Tasks* may, possibly, be distributed over a multiprocessor target architecture, they communicate using message passing and typed ports. A set of 8 communication and synchronization mechanisms is provided (see [19]). Dedicated simulation tools allow to finely validate this design step.

### 3.2 The Event-based Behavior

In a way, a Robot Task is atomic for the decisional level. However it follows an internal sequencing which has not to be seen in normal (failure-free) circumstances. Nevertheless the Robot Task has also to exchange information with the supervisor, described later, which synchronizes and/or conditions their activation. In the present approach, these two aspects are considered in a single way. Thus the Robot Tasks can be considered as reactive systems and can be programmed using the synchrony assumption ([5]): signals are emitted from and to a finite state automaton which specifies the Robot-Task behavior. This automaton, called RTA, is encoded using the synchronous language ESTEREL ([5]).

The signals are emitted by specific module tasks, called "observers". They are strongly typed:

- *the pre-conditions.* Their occurrence is required for starting the servoing task. They may be pure synchronization signals or signals related to the environment, usually obtained through a sensor
- *the exceptions.* They are exclusively emitted by observers in case of failure detection.
- *the post-conditions.* They are either logical synchronization signals emitted by the RTA itself in case of correct termination, or signals related to the environment.

The treatments associated with pre- and post-conditions are quite simple and not described here. The exception processing is more specific:

- *type 1 processing*: the reaction to the received exception signal is limited to the modification of the value of at least one parameter in the module-tasks (gain tuning, for example).

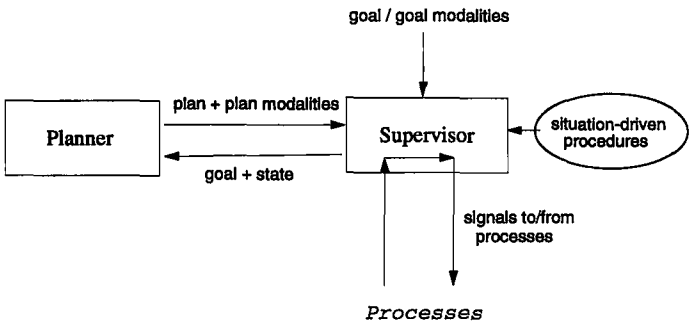


Figure 2: Paradigm for the Integration of Planning and Execution

- *type 2 processing*: the exception requires the activation of a new Robot-Task. The current one is therefore killed. When the ending is correct, the nominal post-conditions are fulfilled. Otherwise, a specific signal is emitted towards the supervisor, which *knows* the recovering process to activate (see later).
- *type 3 processing*: the exception is considered as fatal. Then, everything is stopped.

**Design of a Robot Task:** An object-oriented approach is used for design and modelling of robot-tasks. Based on this approach, a dedicated human-machine interface has been realized. It allows to easily instantiate all the objects required in a given robot task and to specify the values of temporal attributes. Graphic facilities are also provided. Figure 1 gives an example of a robot task which consists in making park a mobile robot. To conclude on this aspect, let us underline that the ESTEREL synchronous code is automatically generated from the object specification. All existing verification tools may then be used on this code without any need for the user to learn the language.

## 4 The Decisional Level

### 4.1 A Paradigm for Integrating Deliberation and Action

We summarize here the paradigm we have adopted in order to take into account the different attributes discussed in section 2.2.

A decisional level is composed of two independent entities: a planner and a supervisor (figure 2).

The *Planner* is given a description of the state of the world and a goal; it produces a plan. One criterion that should be considered when speaking about planning is the "quality" of the produced plan which is related to the cost of achievement of a given task or objective (time, energy, ...), and to the robustness of the plan, i.e., its ability to cope with non nominal situations. This last aspect is one of the motivations of our approach: besides providing a plan, the planner should also provide a set of execution "modalities". These execution modalities are expressed in terms of:
- constraints or directions to be used by a lower plan-

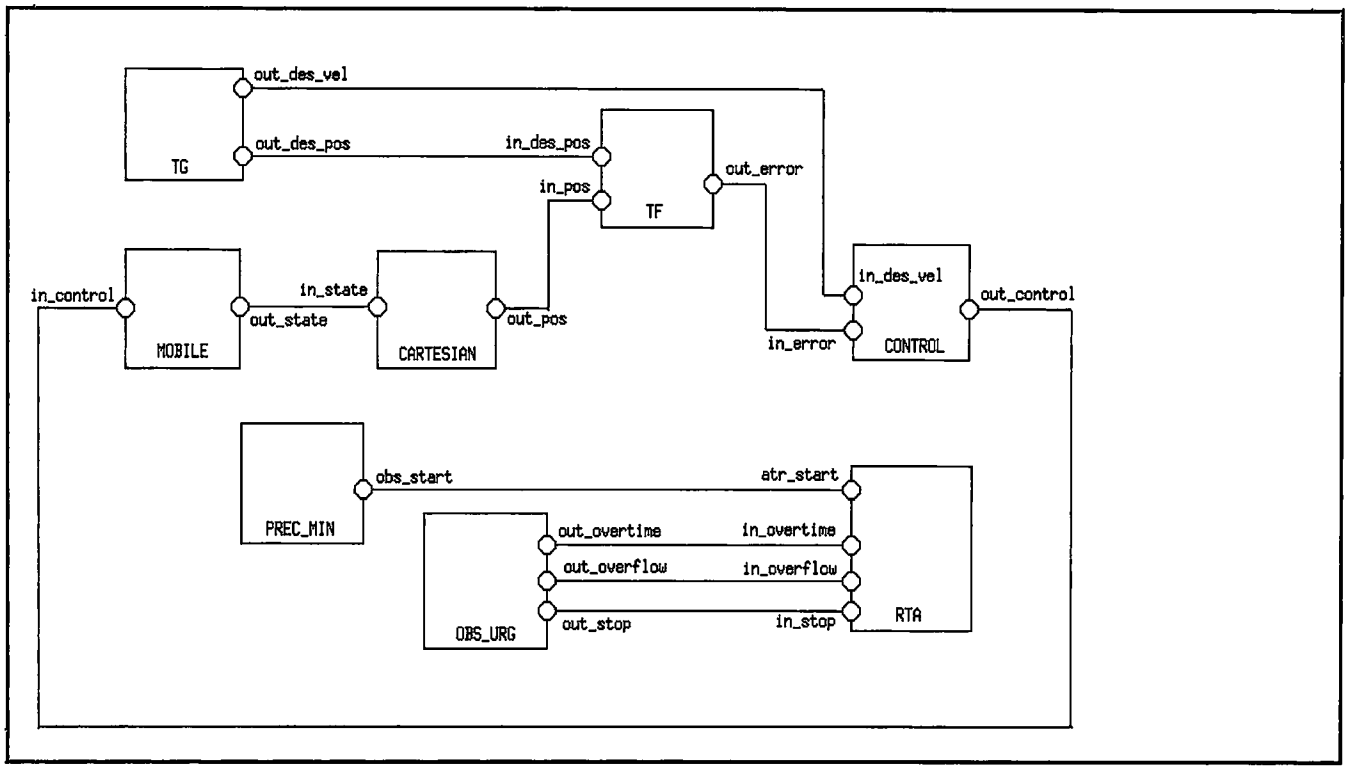


Figure 1: A robot-task example for parking a mobile robot

ning level;
• description of situations to monitor and the appropriate reactions to their occurrence; such reactions are immediate reflexes, "local" correcting actions (without questioning the plan), or requests for re-planning.

These "modalities" provide a convenient (and compact) representation for a class of conditional plans. However, the generation of "modalities" still remains to be investigated: we have no generic method yet for integrating modalities production in a planning algorithm. However, it is possible to produce useful modalities in a second step by performing an analysis of the plan as produced by the a first "classical" planning step. Such an analysis can be based on a knowledge of the limitations of the planner itself and of the world description it uses, as well as on domain or application specific knowledge (see section 5 for example).

The *Supervisor* interacts with the other layers and with the planner. The other layers are viewed as a set of processes which exchange signals with the supervisor. These processes correspond to the actions of the agent as well as events associated with environment changes independent from robot actions.

These processes are under the control of the supervisor which has to comply with their specific features. For example, a process representing a robot motion, cannot be cancelled instantaneously by the supervisor: indeed, such a process has an "inertia". The supervisor may request a stop at any moment during execution; however, the process will go through a series of steps before actually finishing.

The simplest way to represent such processes are finite state automata (FSA). More elaborate representations such as temporized processes should be investigated.

In the FSA we use, at any moment:
• the set of allowed external signals correspond to all the actions that can be taken by the supervisor;
• similarly, the set of possible internal signals correspond to all environment changes that could be perceived by the supervisor.

The activity of the supervisor consists in monitoring the plan execution by performing situation detection and assessment and by taking appropriate decisions in real time, i.e., within time bounds compatible with the rate of the signals produced by the processes and their possible consequences (Figure 3).

The responsibility of "closing the loop" at the level of plan execution control is entirely devoted to the supervisor. In order to achieve it, the supervisor makes use only of deliberation algorithms which are guaranteed to be time-bounded and compatible with the dynamics of the controlled system. Indeed, all deliberation algorithms which do not verify this property are actually performed by the planner upon request of the supervisor.

This execution control is done through the use of the

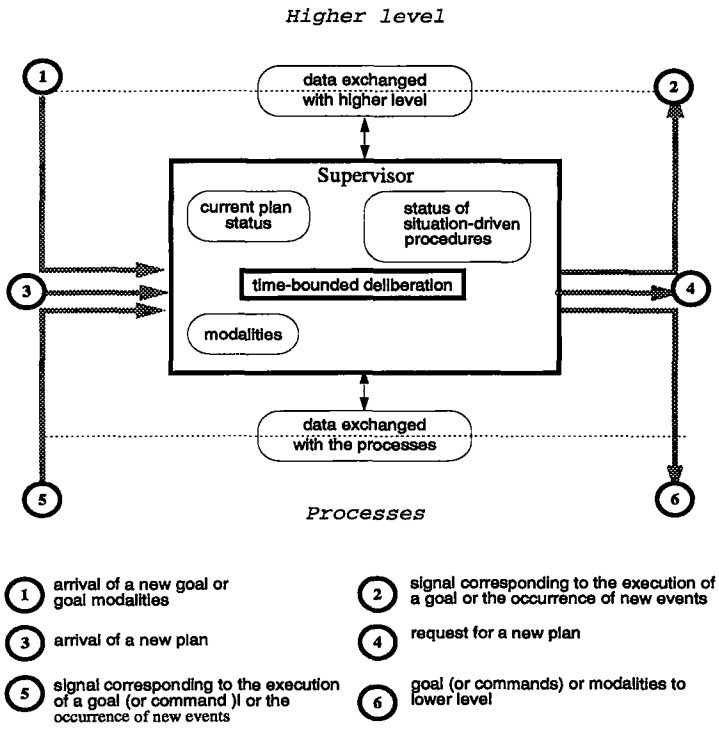


Figure 3: The Supervisor

plan and its execution modalities, as well a set of *situation-driven procedures* embedded in the supervisor and independent of the plan. These procedures are predefined (at design phase), but can take into account the current goal and plan when they are executed, by recognizing specific goal or plan patterns.

## 4.2 A Generic Architecture

We propose here below an architecture which is adapted to a class of autonomous robot applications where it is possible (and suitable) to describe the world model in an abstract symbolic level, and where the robot is given goals expressed in terms of a state. However, even though there is sufficient information to build and maintain such a description, this does not mean that detailed information is also available. Typical cases are "intervention robots" that have to operate in an ill-known environment discovered by the robot's sensors and on which only incomplete and uncertain previous knowledge may exist. Examples are disaster intervention (e.g., the AMR-EUREKA [16] project) or planetary rovers (e.g., VAP[1] [12].

The global system architecture we propose is organized into three levels representing two decisional layers and a functional level (figure 4). The two upper levels are built with the supervisor-planner paradigm. The higher level uses a temporal planner. The second level receives tasks that it transforms into scripts (procedures) composed of elementary robot actions, and supervises script execution while being reactive to asynchronous events and environment conditions. Planning at this level is a "refinement" using domain or task specific knowledge: it does not use a general planner.

[1]VAP: Autonomous Planetary Vehicle, a project of the French national space agency CNES.

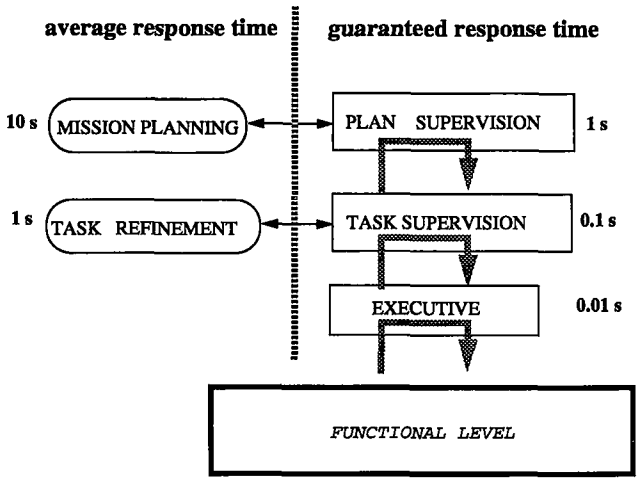


Figure 4: Global Architecture

The functional "execution" embeds a set of elementary robot tasks implementing task-oriented servo-loops as well as a set of robot primitive functions (motion planner, perception, etc...).

# 5 An Instance of the Architecture

We present next an example of the architecture which complies with the different properties discussed in the previous sections.

**At The Mission Planning Level:** For the planner, we have developed a temporal planning system called *IxTeT* (Indexed Time Table) [11, 9] which can reason on symbolic and numeric temporal relations between time instants.

A Plan, as produced by IxTeT, is a set of partially ordered tasks, together with temporal constraints such as minimal and maximal expected durations, and synchronization with expected external events or absolute dates.

In the current implementation, the temporal plan supervisor is provided with *automaton classes* corresponding to the execution of the different actions which may be included in the plan. Whenever an action is started an automaton of the associated class is instantiated. The "internal" events, as well as the "external" events, are represented as time-stamped transitions in the world model as it is used by the temporal planner.

**At the task-planning level:** At this level, we use C-PRS [15] which provides a suitable framework for goal-driven as well as situation-driven deliberation processes. Indeed, PRS implements script (called KA in PRS) se-

lection and goal posting mechanisms. Planning can be performed through context dependent goal decomposition; situation-driven reaction can be performed by triggering KAs according to the world model content.

**At the executive level:** This level is purely reactive with no planning capacities. It controls the execution of robot-tasks and other robot primitive functions using pre-defined context-dependent actions. It is implemented using a rule-based system KHEOPS [10] which allows to compile (off-line) a set of rules, producing a program which performs a time-bounded search through a decision-tree.

**At the functional level:** Concerning the functional level, an example of architecture is described is [19] and an implementation is given in [18]. It allows the user to specify complex robot-tasks with temporal properties, to check their performances in a discrete-time multi-rate framework and on a target hardware architecture. It also provides with automatic code generation for automaton encoding as well as for execution within a real-time operating system.

## 6 Conclusion

We have presented some key design issues of intelligent control architectures for autonomous robots. We have discussed the relevant attributes and showed how they can be integrated in a coherent architecture which provides a framework for implementing time-bounded decision and reaction at different levels, from task-oriented closed loops to situation-driven decision and goal-driven plan generation. We have also briefly presented several tools which can be used to implement instances of this generic architecture.

## References


[1] J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.

[2] J.G. Bellingham, T.R. Consi, "State Configured Layered Control, *Proc. 1st Workshop on Mobile Robots for Sub-sea Environments*, pp 75-80, Monterey, October 1990.

[3] A. Benveniste and G. Berry, "The Synchronous Approach to Reactive and Real-Time Systems", *Proceedings of the IEEE*, vol. 79, no 9, September 1991.

[4] R. F. Camargo, R. Chatila, R. Alami. Hardware and Software Architecture for Execution Control of an Autonomous Mobile Robot. IECON '92, San Diego, USA, Nov. 1992.

[5] G. Berry and G. Gonthier, "The Synchronous Esterel Programming Language : Design, Semantics, Implementation",*Science of Computer Programming*, vol 19, no 2, pp 87-152, Nov. 1992.

[6] R.B. Byrnes, "The Rational Behavior Model: A Multi-Paradigm, Tri-Level Software Architecture for the Control of Autonomous Vehicles", PhD Dissertation, Naval Postgraduate School, Monterey, USA, March 1993.

[7] R. Chatila, R. Alami, B. Degallaix, and H. Laruelle. " Integrated planning and execution control of autonomous robot actions". In *Proc. IEEE Int. Conf. on Robotics and Automation, Nice (France)*, 1992.

[8] E. Coste-Manière, B. Espiau and D. Simon, "Reactive Objects in a Task-Level Open Controller, *Proc. IEEE Conf. on Robotics and Automation*, pp 2732-2737, Nice, France, May 1992.

[9] C. Dousson, P. Gaborit, and M. Ghallab. Situation Recognition: Representation and Algorithms. In *Proceedings of the International Joint Conference on Artificial Intelligence*, Chambéry, France, 1993.

[10] M. Ghallab and H. Philippe. A compiler for real-time Knowledge-based Systems. In *International Workshop on Artificial Intelligence for Industrial Applications*, Hitachi City, Japan, May 1988. IEEE.

[11] M. Ghallab, R. Alami, and R. Chatila. Dealing with Time in Planning and Execution Monitoring. In R. Bolles, editor, *Robotics Research: The Fourth International Symposium*. MIT Press, Mass., 1988.

[12] G. Giralt and L. Boissier. The French Planetary Rover VAP: Concept and Current Developments. IROS'92, pages 1391–1398, July 1992.

[13] G. Giralt, R. Chatila, and R. Alami. "Remote Intervention, Robot Autonomy, And Teleprogramming: Generic Concepts And Real-World Application Cases". IROS'93, Yokohama, japan, July 1993.

[14] A.J. Healey, R.B. McGhee e.a., "Mission Planning, Execution and Data Analysis for the NPS AUV II Autonomous Underwater Vehicle", *Proc. 1st Workshop on Mobile Robots for Sub-sea Environments*, pp 177-186, Monterey, October 1990

[15] Ingrand, F. F., Georgeff, M. P., and Rao, A. S. An Architecture for Real-Time Reasoning and System Control. *IEEE Expert, Knowledge-Based Diagnosis in Process Engineering*, 7(6):34–44, December 1992. Also available as LAAS Technical Report 92-521.

[16] R. Laurette, A. de Saint Vincent, R. Alami, R. Chatila, and V. Pérébaskine. Supervision and Control of the AMR Intervention Robot. pages 1057–1062, June 1991.

[17] F. Noreils and R. Chatila. Control of mobile robot actions. In *IEEE, International Conference on Robotics and Automation, Scottsdale - Arizona*, pages 701 – 712, June 1989.

[18] P. Rives, R. Pissard-Gibollet, K. Kapellos, "Development of a Reactive Mobile Robot using Real-Time Vision" *Third Int. Symp. on Experimental Robotics*, October 28-30 1993, Kyoto, Japan

[19] D. Simon, B. Espiau, E. Castillo, K. Kapellos, "Computer-aided Design of a Generic Robot Controller Handling Reactivity and Real-time Control Issues", *IEEE Trans. on Control Systems and Technology*, to appear, fall 1993