A survey of control architectures for autonomous mobile robots

Adelardo A. D. Medeiros
Laboratório de Engenharia de Computação e Automação (LECA/DEE)
Universidade Federal do Rio Grande do Norte (UFRN)
59072-970 Natal RN Brazil
adelardo@leca.ufrn.br

Abstract

This paper identifies attributes of intelligent robotic applications and surveys the different flavor in robot control architectures. Directions in robot control are discussed and the attributes and properties of different proposals are classified and compared. In the conclusion we present our point of view about the current state of designing robot control architectures.
Keywords: Control architectures; Mobile robots.

1 Introduction

A autonomous mobile robot must be extremely self-reliant to operate in complex, partially known and challenging environments using its limited physical and computational resources. Its control system must ensure in real time that the robot will achieve its tasks despite all these constraints. The control is required to be "reactively" fast but also thorough, while preserving some properties like stability and robustness.

The control and computation architectures should be designed to meet all these requirements. The existing research experience seems to have not yet produced a definitive paradigm for the distribution and/or coordination of the functionalities required for all these kinds of autonomous robots. In the next sections we review the development of robot control architectures, discuss the main approaches and point out some of the major properties of different proposed systems.

2 System requirements

The control system for an non-trivial robot (i.e. a robotic system with a certain degree of autonomy and complexity) is required to meet some behavior specifications and design requirements:

Reactivity to the environment - The vehicle should be reactive to sudden changes in the environment and capable to take into account external events with time bounds that are compatible with the correct, efficient and safe execution of its tasks.

Intelligent behavior - This requires that different compromises be made based on common sense rules in order to exhibit intelligent behavior. The reactions of the robot to external stimuli must be guided by the objectives of its main task.

Multiple sensor integration - The limited accuracy, reliability and applicability of individual sensors must be compensated for by the integration of several complementary sensors.

Resolving of multiple goal - In the case of mobile robots, situations requiring conflicting concurrent actions are inevitable. The control system should provide means to fulfill those multiple goals.

Robustness - The robot must handle imperfect inputs, unexpected events and sudden malfunctions.

Reliability - Is the ability to operate without failures or performance degradation over a certain period.

Programmability - A useful robot should be able to achieve multiple tasks which are described at some abstraction level, instead of only one precise task.

Modularity - The control system of autonomous vehicles should be divided into smaller subsystems (or *modules*) that can be separately and incrementally designed, implemented, debugged and maintained.

Flexibility - Experimental robotics require continuous changes in the design during the implementation phase. Therefore, flexible control structures are required to allow the design to be guided by the success or failure of the individual elements.

Expandability - A long time is required to design,

build and test the individual components of a robot. Therefore, an expandable architecture is desirable in order to be able to build the system incrementally.

Adaptability - As the state of the world changes very rapidly and unpredictably, the control system must be adaptable in order to switch smoothly and rapidly between different control strategies.

Global reasoning - A global high-level decision-making agent, responsible for the understanding of the overall situation, is required to account for the errors introduced by misinterpretation of the sensory data and to fusion the partial available informations.

2.1 Design trade-offs

The level of autonomy as well as task and environment complexity constitutes the design space for intelligent robots. Compromises must be made in order to keep the overall complexity at a reasonable level. Possible trade-offs are:

Reducing the level of autonomy, e.g. by concentrating on routine decisions and time consuming tasks but delegating more difficult tasks or decisions to a human operator.

Reducing environment complexity, e.g. by changing the environment to make it more robot friendly by introducing landmarks for navigation, interfaces for elevators, etc.

Immediacy versus assimilation. At some point, the benefits of data assimilation must be sacrificed to achieve the immediacy needed for real-time response.

3 Proposed architectures

Several control architectures have been proposed. We present four of the more representative ones: the *NASREM* architecture, the *subsumption* architecture, the *TCA* architecture and the *LAAS* architecture.

3.1 The NASREM architecture

The NASREM architecture, proposed by Albus [3, 2], is represented in figure 1. The perceived information passes through several processing stages until a coherent view of the current situation is obtained. After that, a plan is adopted and successively decomposed by other modules until the desired actions can be directly executed by the actuators.

3.2 The subsumption architecture

In the subsumption architecture, developed by Brooks [7], layers of control system are built to let the robot

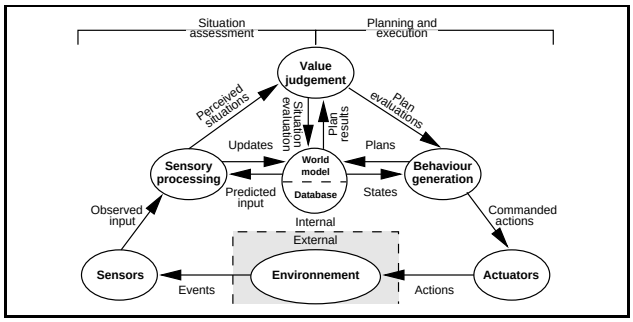

Figure 1: The NASREM architecture

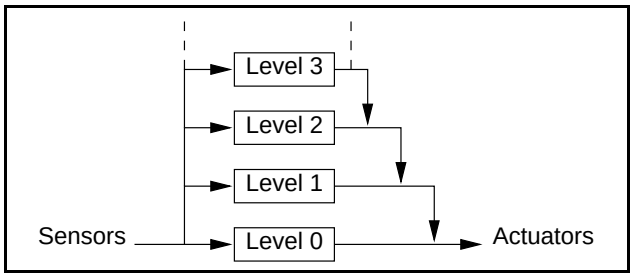

Figure 2: The *subsumption* architecture

operate at increasing levels of competence. Layers are made up of asynchronous modules that communicate over low-bandwidth channels. Each module is an instance of a fairly simple computational machine. Higher-level layers can subsume the roles of lower levels by suppressing their outputs. However, lower levels continue to function as higher levels are added (fig. 2).

Each level of competence includes as a subset each earlier level of competence. Since a level of competence defines a class of valid behaviors, it can be seen that higher levels of competence provide additional constraints on that class. For instance, Brooks used the following four first levels of competence:

0. Avoid contact with objects.
1. Wander aimlessly around without hitting things.
2. "Explore" the world by seeing places in the distance that look reachable and heading for them.
3. Build a map of the environment and plan routes.

3.3 The TCA architecture

The Task Control Architecture, developed by Simmons [25, 26], provides a general framework for controlling distributed robot systems. In essence, TCA is a high-level robot operating system with an integrated set of commonly needed mechanisms to support distributed communications, task decomposition, resource management, execution monitoring and error recovery.

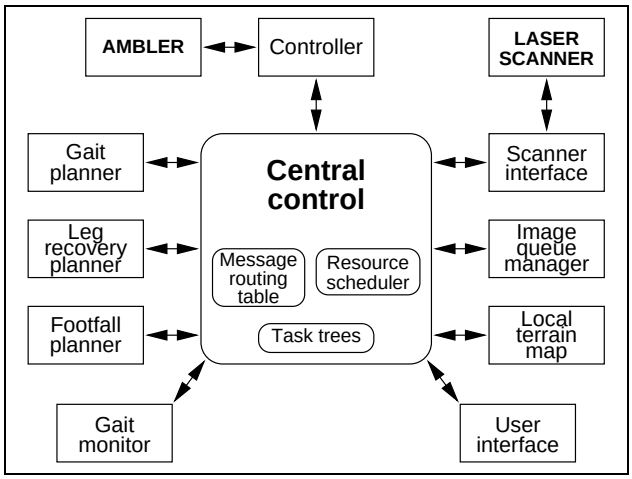

Figure 3: The *TCA* architecture

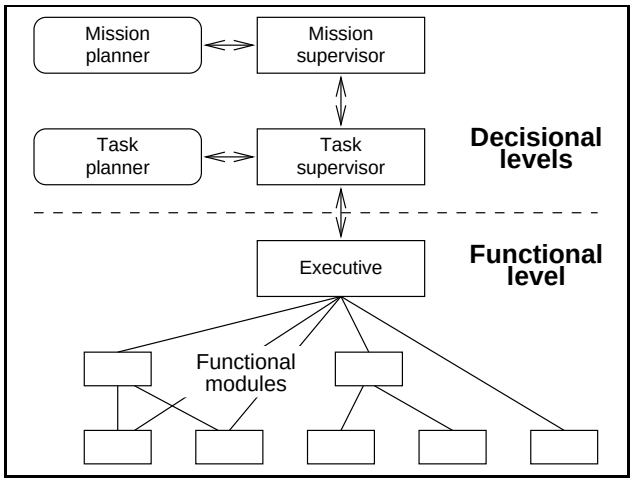

Figure 4: The *LAAS* architecture

A system using TCA consists of a number of task-specific modules and a general-purpose reusable central control module. The modules communicate with one other (and with the central control) by passing messages. The modules register message handling procedures with TCA and the central control has responsibility for routing messages to the appropriate modules. In fig. 3 we can see a example of an robot control system using TCA (the Ambler Walking System).

The TCA architectures retains many concepts of the blackboard architectures [15, 33], but differs from them because while TCA does maintain control information centrally, the actual data needed to solve problems is distributed amongst the system's processes.

3.4 The LAAS architecture

The LAAS architecture [1] is organized in three levels representing two decisional layers and a functional level (fig. 4). The higher level uses a temporal planner, like IxTeT [14]. The second one, usually based on PRS [16], 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. The functional level embeds a set of elementary robot tasks implementing servo-loops and the robot primitive functions (motion planner, perception, etc.)

The functional level is composed of a set of modules [12]. A module embeds primitive robot functions which share common data or resources. This level is managed and controlled by an executive [21]. Modules interact by message passing, by reading data exported by other modules and by putting their own processing results into exported data structures.

4 Main paradigms

Throughout the years of building robot control systems, a number of good practices, design directions and references have been created. Some of the more commonly known paradigms are listed here.

4.1 Sense–model–plan–act (SMPA)

The SMPA paradigm is a nutshell in any control system. The extent to which these activities are instantiated in particular systems varies greatly. Specifically, the amount of deliberation (modeling and planning) can be strongly emphasized or totally neglected.

One of the most direct implementations of the SMPA paradigm is the NASREM architecture (section 3.1). The primary drawback of this approach is that the series of stages through which all sensor data must pass places an unavoidable delay in the loop between sensing and action. The delay can only be reduced by increasing the throughput of the modules.

The SMPA paradigm assumes a quasi-static or at least predictable world assumption between sensing and acting. As this is a rather invalid assumption for most real world domains, approaches have been undertaken to improve reactivity by breaking up the original single SMPA execution pipeline into a number of parallel and interweaved ones. The straightforward ways to accomplish this are vertical and horizontal (or lateral) decomposition.

4.2 Vertical decomposition

Vertical decomposition is based on the assumption that the dynamics of the world decrease with the level of abstraction. The robot control system is vertically

split into levels of hierarchies and the tasks into subtasks of the next lower level. The control flow (initiation and termination of subtasks) is up-down and the data-flow (execution results and sensor readings).

Levels of low abstraction such as servo control are recognized by high immediacy and depend on little or slightly assimilated information. Higher level in contrast may depend on highly symbolic assimilated data. Reasoning on higher levels results in plans with long durations. Some proposal using vertical decomposition are the LAAS architecture (section 3.4) and other hierarchical systems [31].

The reasons for vertical system decomposition are increasing degrees of abstraction, increasing bandwidth and decreasing frequency of interaction with the environment. Drawbacks of purely vertical decomposition lie with the separation of information acquisition and usage, causing loss of efficiency, as well as engineering relates issues, such as the early definition of interfaces between layers reducing expandability.

4.3 Lateral decomposition

Reasons for horizontal, or lateral, decomposition are far more reactive response and simpler modeling of lower level continuous maintenance, achievement and monitoring tasks such as obstacle avoidance.

The lateral decomposition denotes co-operative decentralized problem solving on each level of abstraction. It is based on control entities (usually called behaviors) to horizontally split up the robot control architecture into concurrent operations. Examples for lateral decomposition are the TCA architecture (section 3.3), blackboard-based architectures [15], distributed systems [4] and some behavioral approaches

4.4 Reactive systems

In this context, reactivity connotes either the speed of reaction or minimal usage of internal state information. The "minimal (or no) use of state information" implies that behavior tend to be little or not goal directed. The neglectance of state information, in particular world models, makes it difficult to enable look ahead planning and goal-directed behavior.

Very simple computational units can be used which ensure fast computation and response to external triggers enabling the robot to survive (at last in the short term) in dynamic environments. Different approaches have been developed to create reactive systems on different levels of abstraction.

For the lowest (non-symbolic) level of abstraction, the main proposals are the subsumption architecture (section 3.2), other behavioral systems [19] and some proposals based on neural nets [24] or fuzzy logic. Another important group is based on *situated automata*.

Situated automata are finite state machines which receive sensor input via an update function and have an action function for physical interaction with the environment. The execution cycle consists of reading the sensor input, updating the internal state and the (time-bounded) calculation of the output vector.

Various support tools have been used to create reactive robot systems based on situated automaton formal methodology, like Esterel [6] and Rex [17]. Some other approaches propose specialized tools to design and implement complete robot control systems based on finite state machines: the main examples are Orccad [28], Chimera [30] and ControlShell [23].

For the higher (symbolic) level of abstraction, *reactive planning* and *compiled plans* have been used.

Reactive planning approaches are recognized by the use of plan templates for task decomposition and heuristics to select among different possible instantiations [29]. Plan templates can be very complex and include parallelism and choice. For this reason they are often called *procedural knowledge* [13].

Compiled plan approaches consider all possible situations (or at least a large subset of them) and map appropriate actions to them instead of using explicit run-time planning to form a plan to reach the goal state from the current situation. An example are the *teleo-reactive trees* [22].

5 Classification of systems

As there are many proposed different architectures, it is important to retain the main characteristics of each approach. We postulate that the majority of the architectures found in the literature can be categorized according to three distinct aspects:

- the way their modules are interconnected (hierarchical × centralized architectures);
- the function of the modules (functional × behavioral systems); and
- the way the modules and the environment communicate (reactive × deliberative control).

Of course, not all the proposed control architectures can be easily classed into these categories. Some of them combine reactive and functional modules, some others are half functional, half behavioral, etc. And there is also some ones that we have to consider separately: Sekiguchi and others [24], for instance, proposed the control for a mobile robot by a structured neural network. In this case, as we can hardly identify

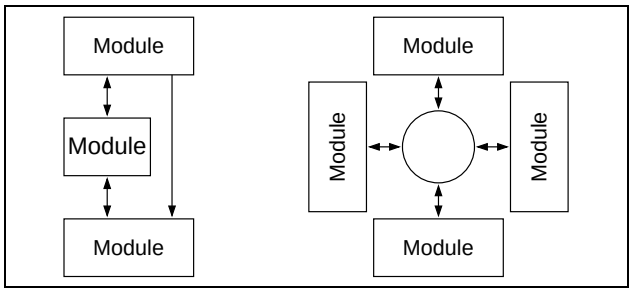

Figure 5: Hierarchical *versus* centralized systems

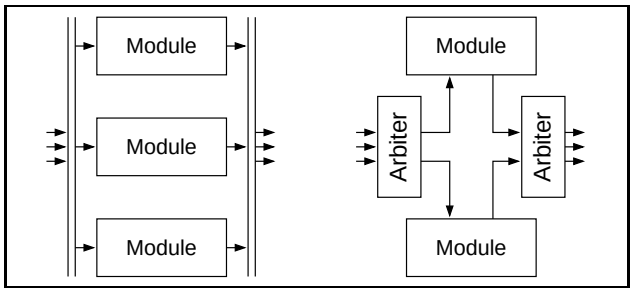

Figure 6: Reactive *versus* deliberative systems

the independent modules, all the previously presented definitions do not apply.

5.1 Hierarchical × centralized

Hierarchical systems have their modules disposed in a particular, *a priori* fixed, structure (frequently in layers or levels). Each module communicate with predetermined others, for instance the neighbor or the lower levels. The modules of centralized systems, in contrast, are disposed around a central module. This central module communicate with all the others and all the other modules communicate with it (fig. 5).

5.2 Functional × behavioral

Each module of a functional control system must provide a service for the system. For instance, we can have modules for perception, modeling, planning, task execution and motor control. In a behavioral system, differently, each module or level directly generate some part of the behavior of the robot: in the purest form of this model each module incorporate its own perceptual, modeling and planning requirements. One module may know how to avoid objects, one other how to build a map of the environment, etc.

5.3 Reactive × deliberative

A reactive system can be defined [5] as a system that maintains a permanent interaction with its environment. In a reactive control system all the modules are connected to the concerned sensors and/or actuators, whereas in a deliberative system there is a policy of access to the environment carried out by an specialized arbiter, like a planner (fig. 6).

5.4 Other proposed definitions

Our classification of control architectures is neither unique nor universal. Different definitions of terms like behavioral, reactive and centralized architectures have been proposed. See, for example, the definitions of Fayek and others [11], Simmons [27] and Brooks [9]. We can speculate that some of these terms are at times considered to be synonyms mainly because:

- Despite of the works in *reactive planning*, almost all the functional systems are deliberative. In fact, it is extremely difficult to design high-level functional modules (as planners) sufficiently thorough to directly interact with the environment.
- Similarly and for the same reasons, almost all the reactive systems are behavioral.
- The majority of the behavioral systems is reactive. Lewis and others [18], nevertheless, put forward a not completely reactive behavioral architecture for a flying vehicle. The high-level behaviors do not directly affect the outputs to the actuators, but instead modulate the low-level behaviors.
- The majority of the functional systems is hierarchical. But we can find some examples of functional centralized systems, like the one proposed by Simmons (section 3.3).
- The majority of the centralized systems is deliberative and the majority of the reactive systems is hierarchical. Fayek and others [11], however, proposed an architecture based on reactive behavioral modules and a blackboard central unit to select and affect the parameters of the appropriate module to handle the situation in hand.

We can compact these statement using $\rightarrow$ for "almost surely implies" and $\rightsquigarrow$ for "usually implies":

$$\begin{array}{rcl} \text{Behavioral} & \overset{\rightsquigarrow}{\leftarrow} & \text{Reactive} \\ \text{Centralized} & \rightsquigarrow & \text{Deliberative} \\ \text{Functional} & \rightarrow & \text{Deliberative} \\ \text{Functional} & \rightsquigarrow & \text{Hierarchical} \\ \text{Reactive} & \rightsquigarrow & \text{Hierarchical} \end{array}$$

6 Main characteristics

One of the liveliest debates in the domain of the control systems for autonomous robots is about the behavioral *versus* functional architectures. The supporters of the behavioral approach, as in [18], argue that living systems have evolved patterns of behavior which require neither explicit models for control nor detailed planning before they can act. An insect's nervous system does not contain explicit mathematical representations of the dynamics of its body, wings and legs.

Brooks says that "traditional" robotics and artificial intelligence seem unable to deliver real-time performance in a dynamic world. Behavioral systems can seem to have more limited abilities, but he argues that the proposed functional systems operate in a way that will never be transportable to the real world [9]. Brooks has also stated [8] that this style of computation can break the Von Neuman bottleneck that throttles computation of information-rich sensory data.

Those who espouse the functional approach, nevertheless, present different arguments. Simmons [26] says that one problem with the behavioral approach is that it is extremely difficult to get the primitive behaviors exactly right from the start, hence some augmentation is needed when previously unforeseen situations arise. As the higher-level behaviors are designed supposing known the characteristics of the lower-level ones, a modification to a primitive behavior may entail modifications to several other behaviors.

Tsotos [34] argues that the *strict* computational behaviorist position for the modeling of intelligence does not scale to human-like problems and performance. He postulates that the ability to search images to find stimuli on which to act is a necessary component of any intelligent agent. If targets are not explicitly known and used to optimize search[1], he shows that the problem is NP-hard. Experimental results seem to agree with this opinion, since most of the strictly behavioral robots are not based on complex sensors.

An other criticism of the behavioral approach is that, as this kind of robot does not have an action plan, its global behavior is constructed from the interactions between the limited behaviors. We can find some opinions [10] supposing these interactions are insufficient for manifesting more evolved behaviors. A behavioral control systems would be capable of supervising a robot with predefined tasks, not a programmable robot.

This discussion about functional or behavioral architectures sometimes subsumes the debate about the other aspects of control systems: the reactive or deliberative approach and the centralized or hierarchical organization. When considering these two other features of control systems, usually the main problem is to find the right equilibrium between the desired "intelligent behavior" and the necessary reactivity, since:

- The existence of an specialized agent deliberating about the access to the environment usually improves the quality of the robot's response to changes in the surroundings. But it can also introduce time delays between the decision and the execution of the actions.
- A central module with a global knowledge of the robot generally leads the system to make better choices about the actions to accomplish, but it is a potential communication bottleneck.

Therefore, adopting some of these options is a technological problem where the designer must consider the required degrees of reactivity and "intelligent" behavior and the related implementation cost. According to the kind of robot and the necessary level of autonomy, we can find in the literature different recommendations about the more appropriate architecture.

7 Conclusions

As remarked by Simmons [27], in many cases different architectures can be used for the same tasks. Typically, the differences between architectures are the ease with systems can be developed and the efficiency with tasks can be achieved. For instance, if you use a reactive control systems, the reactivity of the robot can be easily obtained, but some work is necessary to be sure the actual robot behavior corresponds to the desired one in all situations. On the other hand, with a deliberative control system the robot responses are more predictable, but an excessive complexity of the policy of access to the environment can dangerously increases its reaction time.

Many authors [11, 12, 27, 32] advocate the combination of different paradigms. Several architectures, called hybrid architectures, have been designed to overcome problems inherent in traditional deliberative architectures and reactive architectures by blending deliberative planning, i.e. symbolic representation and reasoning, and reactive mechanisms in order to overcome their disadvantages.

7.1 Hybrid architectures

Three layer architectures seems to be the current state of evolution. Three layer architectures employ three

[1] Knowledge of the target is explicitly forbidden in the strict interpretation of the published behaviorist dogma.

levels of abstraction: the deliberative layer, a sequencing layer and a reactive layer.

Activities on the deliberative layer correspond to long term strategic planning as well as eventual plan adaptations. This level relies on very abstracted knowledge, highly sophisticated reasoning techniques and is the typical domain of AI planners able to make use of extensive application domain knowledge.

The sequencing layer involves a reactive planner which selects and executes appropriate tactics using context dependent rules. A tactic is a pre-written ordered set of actions, rich in structure, i.e. disjunction and conjunction, recursion and hierarchies of actions. Execution of tactics finally leads to activation and termination of reactive layer behaviors.

The reactive level performs the transition from symbolic reasoning to non-symbolic numerical control and the combination of separate behaviors.

Within three layer architectures, communication and interaction is of greatest concern. The general approaches are:

- **Deliberative layer – sequencing layer**: The connection is done by mapping tactics to operator descriptions. Sequencing layer tactics are represented by planning operators in the deliberative layer. Parameters are bound by unification.
- **Sequencing layer – reactive layer**: Interaction between these two layers involve (a) activation; (b) deactivation or termination; (c) passing of control parameters; and (d) monitoring of success, as well as conversion of non-symbolic to symbolic parameters and vice-versa.

The LAAS architecture (section 3.4) is an representative instance of hybrid control systems based on several paradigms. The different agents to sense (perception modules), model (mapping modules), plan (planners) and act (actuator modules) indicates the observance of the SMPA paradigm. The hierarchy of levels implements a vertical decomposition. The modules, under certain circumstances, can operate in parallel to directly access predefined sensors and actuators, typifying a lateral decomposition of reactive sub-systems.

An advantage of hybrid architectures is that, according to the considered application, some levels of the hierarchy can be suppressed. In [20] we can see an implementation without the higher planning level, unnecessary because of the small duration of the task. This example demonstrates the communication and interaction protocols between the modules, an executive and the decisional level in a real world application.

References

[1] R. Alami, R. Chatila, S. Fleury, M. Ghallab and F.F. Ingrand. **An architecture for autonomy**. To appear on *International Journal of Robotics Research*[e], Spring 1998.

[2] J.S. Albus. **Outline of a theory of intelligence**. *IEEE Trans. on Systems, Man and Cybernetics*, 21(3):473–509, May 1991.

[3] J.S. Albus, H.G. McCain and R. Lumia. **NASA/NBS standard reference model for telerobot control system architecture (NASREM)**. Technical Report 1235, National Institute of Standards and Technology, 1989.

[4] P. Baroni, G. Guida, S. Mussi and A. Vetturi. **A distributed architecture for control of autonomous mobile robots**. In *ICAR*[c], Sant Feliu de Guixols, Spain, Sept. 1995.

[5] A. Benveniste and G. Berry. **The synchronous approach to reactive and real-time systems**. *Proc. of the IEEE*, 79(9):1270–1282, Sept. 1991.

[6] F. Boussinot and R. de Simone. **The ESTEREL language**. *Proc. of the IEEE*, 79(9):1293–1304, Sept. 1991.

[7] R.A. Brooks. **A robust layered control system for a mobile robot**. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, Mar. 1986.

[8] R.A. Brooks. **Intelligence without reason**. In *IJCAI*[d], Sydney, Australia, Aug. 1991.

[9] R.A. Brooks. **New approaches to robotics**. *Science*, 253:1227–1232, Sept. 1991.

[10] R. Chatila. **Deliberation and reactivity in autonomous mobile robots**. *Robotics and Autonomous Systems*, 16:197–211, Dec. 1995.

[11] R.E. Fayek, R. Liscano and G.M. Karam. **A system architecture for a mobile robot based on activities and a blackboard control unit**. In *ICRA*[a], Atlanta, USA, May 1993.

[12] S. Fleury, M. Herrb and R. Chatila. **Design of a modular architecture for autonomous robots**. In *ICRA*[a], San Diego, USA, May 1994.

[13] M.P. Georgeff and A.L. Lansky. **Procedural knowledge**. *Proc. of the IEEE*, 74(10): 1383–1398, Oct. 1986.

[14] M. Ghallab and A. Mounir Alaoui. **Managing efficiently temporal relations through indexed spanning trees**. In *IJCAI*[d], Detroit, USA, Aug. 1989.

[15] B. Hayes-Roth. **A blackboard architecture for control**. *Artificial Intelligence*, 26(3):251–321, July 1985.

[16] F.F. Ingrand, R. Chatila, R. Alami and F. Robert. **PRS: A high level supervision and control language for autonomous mobile robots**. In *ICRA*[a], Minneapolis, USA, Apr. 1996.

[17] L.P. Kaelbling and N.J. Wilson. **Rex programmer's manual**. SRI International, Menlo Park, California, USA, 1988. Technical Report 381R.

[18] M.A. Lewis, A.H. Fagg and G.A. Bekey. **The USC autonomous flying vehicle: an experiment in real-time behavior-based control**. In *ICRA*[a], Atlanta, USA, May 1993.

[19] M.J. Mataric. **Integration of representation into goal-driven behavior-based robots**. *IE EE Trans. on Robotics and Automation*, 8(3):304–312, June 1992.

[20] A.A.D. Medeiros and R. Chatila. **Priorities and data abstraction in hierarchical control architectures for autonomous robots**. In *Proc. of Workshop on Intelligent Robotics (Congress of SBC)*[f], Brasília, Brazil, Aug. 1997.

[21] A.A.D. Medeiros, R. Chatila and S. Fleury. **Specification and validation of a control architecture for autonomous mobile robots**. In *IROS*[b][f], Osaka, Japan, Nov. 1996.

[22] N.J. Nilsson. **Teleo-reactive programs for agent control**. *Journal of Artificial Intelligence Research*, 1:139–158, Jan. 1994.

[23] S.A. Schneider, V.W. Chen and G. Pardo-Castellote. **The ControlShell component-based real-time programming system**. In *ICRA*[a], Nagoya, Japan, May 1995.

[24] M. Sekiguchi, S. Nagata and K. Asakawa. **Behavior control for a mobile robot by a structured neural network**. *Advanced Robotics*, 6(2):215–230, 1992.

[25] R.G. Simmons. **Concurrent planning and execution for autonomous robots**. *IEEE Control Systems Magazine*, 12(1):46–50, Feb. 1992.

[26] R.G. Simmons. **Monitoring and error recovery for autonomous walking**. In *IROS*[b], Raleigh, USA, July 1992.

[27] R.G. Simmons. **Structured control for autonomous robots**. *IEEE Trans. on Robotics and Automation*, 10(1):34–43, Feb. 1994.

[28] D. Simon, B. Espiau, E. Castillo and K. Konstantinos. **Computer-aided design of a generic robot controller handling reactivity and real-time control issues**. *IEEE Transactions on Control Systems Technology*, 1(4), Dec. 1993.

[29] M.G. Slack. **Navigation templates: Mediating qualitative guidance and quantitative control in mobile robots**. *IEEE Trans. on Systems, Man and Cybernetics*, 23(2):452–466, Mar. 1993.

[30] D.B. Stewart and P.K. Khosla. **Rapid software development for robotics applications using component-based real-time software**. In *IROS*[b], Pittsburgh, USA, Aug. 1995.

[31] S. Suzuki, J. Iijima and S. Yuta. **Design and implementation of an architecture of autonomous mobile robots for experimental researches**. In *ICAR*[c], Tokyo, Japan, Nov. 1993.

[32] C.E. Thorpe and M. Hebert. **Mobile robotics: Perspectives and realities**. In *ICAR*[c], Sant Feliu de Guixols, Spain, Sept. 1995.

[33] J.Y. Tigli, M. Occello and M.C. Thomas. **Toward a new intelligent reactive controller for autonomous mobile robots**. In *ICRA*[a], Atlanta, USA, May 1993.

[34] J.K. Tsotos. **Behaviorist intelligence and the scaling problem**. *Artificial Intelligence*, 75:135–160, 1995.

[a] Proc. of IEEE Int. Conf. on Robotics and Automation
[b] Proc. of IEEE Int. Conf. on Intelligent Robots and Systems
[c] Proc. of Int. Conf. on Advanced Robotics
[d] Proc. of Int. Joint Conf. on Artificial Intelligence
[e] Available on `http://www.laas.fr/~felix/publis/`
[f] Available on `http://www.leca.ufrn.br/~adelardo/`