Articolo presentato al "SPIE's 1987 Cambridge Symposium on Advances in Intelligent Robotic Systems" - Boston, novembre 1987.

Behavioral Model Architectures: a New Way of Doing Real-time Planning in Intelligent Robots.

Riccardo Cassinis, Ernesto Biroli, Alberto Meregalli, Fabio Scalise

Politecnico di Milano, Dipartimento di Elettronica,
Piazza Leonardo da Vinci, 32, I-20133 Milano - Italy

ABSTRACT

Traditional hierarchical robot control systems, although well suited for manufacturing applications, appear to be inefficient for innovative applications, such as mobile robots.

The research we present aims to the development of a new architecture, designed to overcome actual limitations. The control system was named BARCS *(Behavioral Architecture Robot Control System). It is composed of several modules, that exchange information through a blackboard.*

The original point is that the functions of the modules were selected according to a behavioral rather than a functional decomposition model. Therefore, the system includes, among other, purpose, strategy, movement, sensor handling and safety modules.

Both the hardware structure and the logical decomposition allow a great freedom in the design of each module and of the connections between modules, that have to be as flexible and efficient as possible.

In order to obtain an "intelligent" behavior, a mixture of traditional programming, artificial intelligence techniques and fuzzy logic are used, according to the needs of each module. The approach is particularly interesting because the robot can be quite easily "specialized", i.e. it can be given behaviors and problem solving strategies that suit some applications better than other. Another interesting aspect of the proposed architecture is that sensor information handling and fusion can be dynamically tailored to the robot's situation, thus eliminating all time-consuming useless processing.

Some experimental results are presented.

1. - INTRODUCTION

As robotics are moving from the classical industrial manufacturing field towards other applications, it becomes clear that the control structures that were employed so far are not suitable for driving the new generation of robots.

The authors are mainly concerned with the severe limits of nowadays robot technology, and are fairly convinced that no significant improvements can be obtained unless the global robot philosophy is radically changed.

After an initial stage, where no formal methods had been developed yet, industrial robots have always been programmed exactly as computers were, with a description of the actions to be performed in order to achieve a predefined task [6]. In traditional (*explicit*) programming, the process of finding out the operations to be performed and their correct sequence is left to the programmer, that must develop an algorithm that solves the problem [2, 4]. In *implicit* programming systems this task is (or should) be automatically performed by the machine, but the result is always an explicit program, that contains full information about each elementary action the robot must perform [1]. In other words, implicit programming systems are planners that substitute the human programmer as far as the process of finding the solution algorithm is concerned. No changes (or very small changes) are required on the robots and on their control systems.

So, while algorithmic methods are well suited for programming classical industrial robots, many problems arise if they have to be applied to other robots, and in particular to mobile robots. The ultimate aim of this paper is to show how mobile robots should be built and programmed using a totally different philosophy, that would greatly enhance their capabilities.

2. - PROBLEM'S OVERVIEW

If a traditional robot system is examined with the aim of identifying where the main functions of the system are executed, it will become clear that the machine does not know what it is doing, because the four functions *purpose*, *strategy*, *safety* and *behavior* are not located inside it, but in the brain of the programmer. Such a machine is completely unable of withstanding unforeseen situations, since, not knowing the *final goal* it has to reach, it will never be able to produce alternate plans for reaching it.

In more sophisticated programming systems (*implicit* programming systems) some of the functions previously considered as pertaining to the human programmer are moved inside the machine [5, 7, 10], but the problem of solving emergency situations still remains. In some attempts to make robots capable of generating emergency recovery plans the structure of to-day's programming systems requires that the source program be analyzed in order to extract semantic information from it [11]. This is exactly the same information the programmer had in mind when he wrote the program, and that was lost during the coding process.

As far as mobile robots are concerned, similar problems can be found. Most "toy" robots have a control system and a programming structure that closely resembles the one currently used in industrial robots. But, since their working conditions are very different, their behavior is not at all satisfactory. Research robots (and some advanced industrial models) are on the other hand much more sophisticated, and exhibit very good skills [3, 14, 15]. Nevertheless, in order to allow inferential systems to plan their actions, they often require a huge amount of information to be processed before they can even start the planning phase. For instance, they usually need a complete knowledge of the environment they are in, in order to be able to calculate the path to be followed to reach the goal (navigation problem) [9, 12, 13].

The human approach to a navigation problem is completely different: in order to reach a position that has already been identified, a human will only roughly map the environment in the area that will supposedly be interested by his movements, and will produce an approximate plan (path to be followed and actions to be done). This plan will be refined as the job proceeds, and alternate plans will be generated only if difficulties arise.

Now, if (as it happens with mobile robots) each job will be executed only once, it is not worth spending time for generating a complete and accurate plan before starting the execution.This also applies to repetitive tasks, since in an unstructured environment each iteration of the same task may require a completely different set of actions, and can therefore be considered as the execution of a new program.

The absence of parallel problem solving is another gap: it usually takes a long computational time to perform a decisional analysis when a sophisticated control is needed.

Most existing architectures also exhibit a lack of flexibility in knowledge management and rules application, and have quite poor knowledge description methods

3. - *BARCS* OPERATING PRINCIPLES

In order to overcome at least some of the previously mentioned problems *BARCS* (Behavioral Architecture Robot Control System) Project was started.

BARCS architecture includes several actors that work in parallel and communicate through a blackboard. Each *BARCS* module is specialized for a particular job. Each module contributes to the generation of the robot's behavior, that will lead to the solution of the assigned problem.

3.1. - *BARCS* concepts

BARCS model is based on several parallel activities, such as task decomposition, sensor data analysis, execution and safety control, etc. Such activities are performed by different modules of *BARCS* structure.

Each module is devoted to a specific function; all the modules work in an independent and parallel way.

The conceptual center of this architecture is a sort of blackboard, through which all information must pass, and in which all the data about the robot and the surrounding environment are stored. This also implies that that all information contained in the system is available to any module.

Furthermore, a module must be able to know what the other modules are doing. This is a very important feature of *BARCS*: for instance, the sensory module must always provide the kind of information which is actually required for allowing the other tasks to perform their actions.

Another innovative characteristic of *BARCS* structure is that all the modules are conceptually at the same level; no fixed hierarchies are established.

3.2. - Faced problems

During the development of *BARCS* some general problems were examined, that in our opinion are strictly connected with an innovative concept of robot control structure.

The sensor data management is probably the most critical problem in today's robotics: the physical acquisition of data is generally not simple, if not only simple data, but also more complex information kinds are needed.

Therefore, all the modules must be able to manage information at a symbolic level; only the sensor module knows the physical acquisition procedures for all the data that should be made available to the other modules.

Furthermore, in order to increase the flexibility and the efficiency of the data acquisition, each kind of sensory information should not be related to any particular sensor: an intelligent management of many simple and different sensor data and of their fusion should be implemented.

The use of symbolic representation, that leaves out as much quantitative information as possible, is a good choice not only for data management, but also for knowledge representation and for the syntax of any communication, that takes place among the various modules.

3.3. - Uncertainty management

Uncertainty is treated in several ways and at different levels in *BARCS* model: in the real world representation, in the knowledge base management and in the computation algorithms.

Due to the particular nature of the environment and of the tasks that will be assigned to the robot, it is clear that traditional computation algorithms will not be able to generate actions and behaviors that can face the dynamics of the real world.

It is then necessary to introduce in the decisional analysis some factors that allow to perform "weighted" control actions, whose weights are a function of the particular environment in which the rules are activated.

According to previous consideration, while in classical decision systems we use a selection in mutual exclusion, in *BARCS* structure we aim to activate different rules in a parallel way, and the weighted solution is a fusion of different directives.

4. - *BARCS* STRUCTURE

The logical structure of *BARCS* modules is shown in Fig. 1.

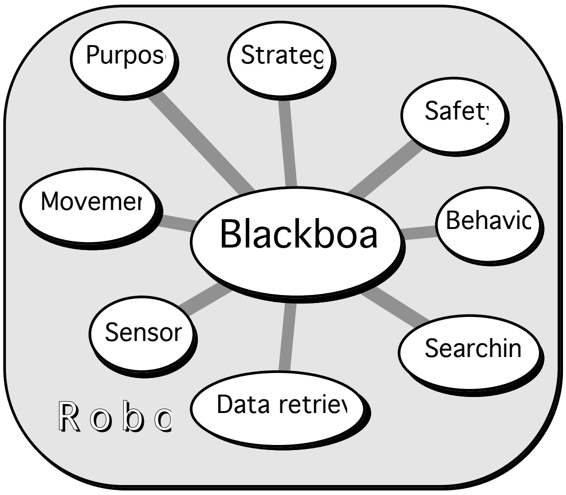

Figure. 1. - *BARCS* structure.

A short description of each module will be given in the sequel. A more detailed description can be found in [8].

4.1. - The blackboard

The blackboard has two functions in *BARCS* structure: it is the system data base, and is used as the only communication path among different modules.

In order to accomplish these two functions, it is implemented as a large structured memory connected to a controller. This controller probably is the most critical part of *BARCS* structure [16]. It must efficiently perform these two functions:

- Handle and queue requests coming from the other modules;
- Handle and maintain the data base.

The first function can be implemented using standard operating systems techniques; the second one is much more difficult, because it implies some "intelligence" in the controller. The problem is that most of the data that will be stored in the blackboard are a model of the real world the robot is in, as it appears to the sensors. Now, the sensors cannot be continuously used to update this model, because some of the operations are time consuming, and can only be done at certain intervals. The task of the controller is then to understand which data are reliable at any time, depending on their kind and on the past actions of the robot. Furthermore, when new data arrive, the controller must decide if they should be stored in new areas, or if they will better overwrite previously stored and now useless (or wrong) data.

4.2. - The purpose and strategy modules

The main purpose of these modules is to define and manage the high level strategy.

The purpose module contains the goal the robot must pursue. Since *BARCS* can not be programmed in a traditional sense, but contains *hardwired* purposes, the main function of this module is just to select the right purpose at the right time.

But complex tasks can be decomposed in a number of simpler tasks, in different ways, according the environmental situation and robot state. This is a job of the strategy module, that can be divided in two different parts:

- A knowledge base that contains all possible strategy decompositions and the bonds of every strategic action;
- An inferential engine that can choose the best decomposition of the current task among the different ones that exist in the data base.

Task decomposition can be either vertical or horizontal; in other words, sub-tasks can be executed in sequence or in a parallel way.

Any task decomposition is associated, in the data base, to a coefficient that defines the (its) efficiency or reliability degree. This coefficient is useful when a new planning phase, or an alternate decomposition becomes necessary.

In this case, decompositions that are alternatives are chosen in function of their degree of confidence expressed by the coefficient.

In conclusion we can consider the case of a complete or a partial new planning, due to a failure in the decomposition that is actually executing.

In such a situation it is not necessary abort the command, because the Purpose module is able to take an alternative way to reach (perform) the task whose last planning was unsuccessful.

4.3. - Behavior and safety modules

Both modules were introduced to control the behavior of the robot for all those situations where no rules are given in the strategy module. For instance, if the robot's purpose is to move an object from one place to another one, the strategy module will provide a list of actions that allow reaching the final goal (locate the object, reach it, grasp it, etc.), but will not contain any suggestion for, say, avoiding a moving obstacle that gets in the robot's way. Therefore, these two modules take care of handling all those situations that are not explicitly foreseen in the strategic decompositions.

Both modules work in the same way, except that the safety module monitors and handles all the situations that can jeopardize the robot's safety or the safety of the surrounding environment, while the behavior module handles the problems that, although not dangerous, do not allow reaching the goal in a straightforward way. Moreover, the behavior module contains and enforces some rules that determine the "social life" of the robot, with respect to the environment and to other robots.

Each elementary action the robot has to do is checked prior to execution for consistency with the actual conditions of the environment. During the execution another control is made, that checks possible changes of the environment, and adapts the instruction parameters to such changes.

Logical interactions among the modules that are relevant to the "planning" activity of the robot are shown in Fig. 2.

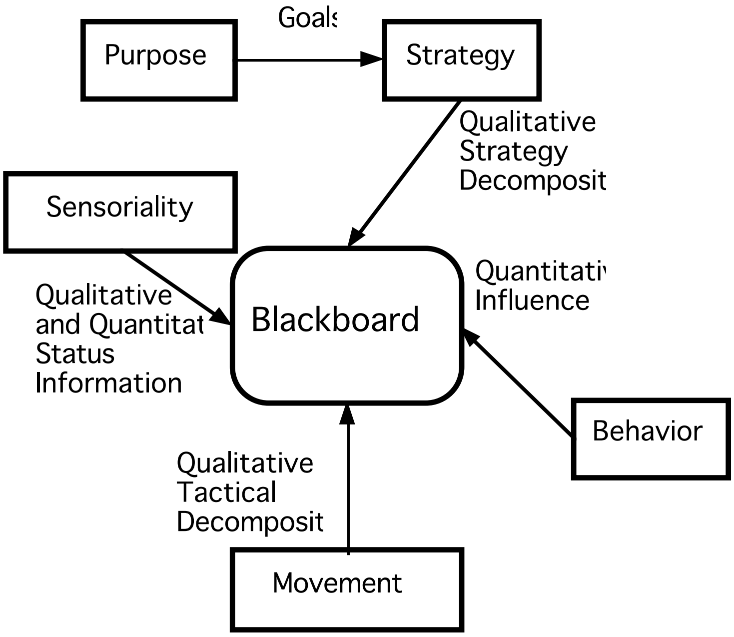

Figure 2. - Logical interactions among modules.

4.4. - The sensing module

The sensing module takes care of handling sensors, and of coordinating and processing the information they provide. Its main task is to understand what kind of information the other modules need, and to gather this information from the various sensors the robot is equipped with. The module also contains the knowledge that allows the acquisition of sensory information needed for identifying and locating objects, that !e≤ÁÔ▯√à¿ includes not only sensors handling, but also physical actions to be performed by the robot (for instance, if an object cannot be "seen", the robot should wander around until it finds it). The complexity of this knowledge obviously depends on the kind of objects to be searched and of the data to be interpreted. In conclusion, the blackboard should always contain up-to-date sensory data that can be readily used by the other modules. The kind of these data depends on the particular goal (or subgoal) being pursued at each moment.

4.5. - The motion module

This module controls all movements of the robot, according to the requests of the other modules and is able to manage a number of requests that are over and have different priority.

This is another critical point in most mobile robots. Usually, motion is considered to be exact (motors are equipped with position and velocity sensors, and robot movements are commanded in terms of exact quantities). Now, we know by experience that the absolute position of a mobile robot is very difficult to compute, unless very expensive systems are used. But, in the general philosophy of *BARCS*, no exact quantities are known in the system: everything is approximate and, in any case, all quantities are related to the goal to be reached, and not to an absolute reference system. Therefore, since even a small error would make absolute calculations impossible or meaningless, it was decided to eliminate all position and velocity systems in *BARCS*, and to use existing sensors to correct the robot's movements as errors are discovered.

Actions of the movement module can be interrupted, if other modules (typically safety and behavior) require urgent tasks to be performed.

5. - SIMULATION AND EXPERIMENTAL RESULTS

For checking the structure's efficiency, simulation of the most critical parts of the system is being carried on.

In order to make simulation software as simple as possible, the parallel architecture of *BARCS* was replaced with a set of modules running on the same computer. A scheduler simulates parallel execution, using traditional time-sharing techniques. This simulation is mainly devoted to testing and refining data structures and communication among modules.

At the same time, *BARCS* structure is being implemented on a very simple mobile robot, whose main task is to demonstrate the feasibility of these concepts. Therefore, it does not have at this moment any special tooling. The robot (Fig. 3) was named *RISK* (Robot In Shape of Kettle), because of its quite odd appearance.

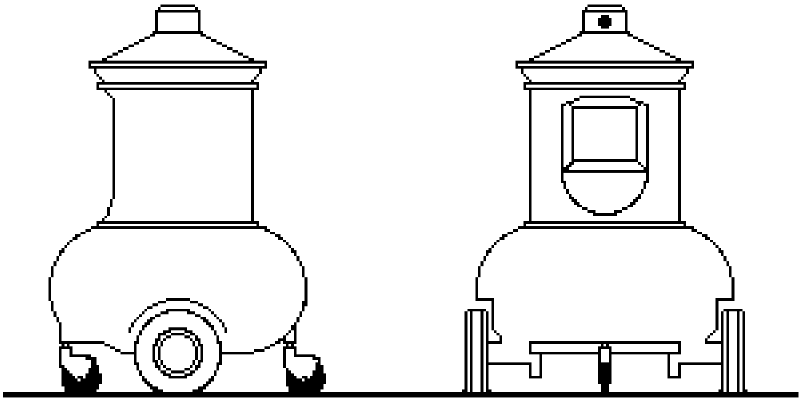

Figure 3 - *RISK* robot.

RISK electronics consist of a Macintosh Plus computer, that is connected to a number of microcomputers (three in the actual version), that drive motors and sensors.

RISK is equipped with several sensors, the most important being a sonar for measuring the distance of obstacles, photodiodes to detect lamps that are used as an aid to the robot for finding and identifying its targets, and a TV camera that allows recognition of simple objects.

The first task that was chosen to demonstrate the capabilities of the robot is the following: the robot must move in an a-priori unknown household environment., where several plants are present, each one equipped with a special device, that glows an infrared light emitting diode when the plant needs watering. The robot's task is to wander around, looking for plants that need watering, and when it locates one, to reach it and water it.

RISK hardware is now complete (Fig. 4), and software is being developed.

Figure. 4. - *RISK* as it is now (October, 1987).

The first experimental results are very encouraging: they show that the behavioral approach just described can be very useful for mobile robots, and that, once the general behavior and safety rules have been established, it is very easy to build the description of different tasks.

6. - CONCLUSIONS

It was shown how traditional robots are intrinsically limited in their performance, and how a non-traditional control structure may be much better suited to drive robots of the next generation.

A structure was proposed, that should overcome problems of non-structured environments and difficult to describe tasks.

The experimental realization of this structure, and the related implementation problems, were discussed, and the experimental results obtained so far were reported.

BIBLIOGRAPHY

[1] ESPRIT Project No. 623: "Operational Control for Robot System Integration into CIM: System Planning, Implicit and Explicit Programming", 4th Interim Report, IPK, Berlin (1987).

[2] Bison,P., Lorenzin,G., and Pagello, E.: "The Formal Definition of VML and a Proposed Portable Implementation", Proc. XI ISIR, Tokyo (1981).

[3] Brooks, R.A.: "A Robust Layered Control System for a Mobile Robot", IEEE Journal of Robotics and Automation, March, (1986).

[4] Cassani, F., Cassinis, R., Manes, A., and Miracola, G.: "HVML: High Level Virtual Machine Language - Preliminary Definition", ESPRIT P623 Project, Working Paper IP - PdM - 4.87/1, Milano (1987).

[5] Cassinis, R.: "Automatic Resource Allocation in Industrial Multirobot Systems", Proc. 1986 IEEE Conference on Robotics and Automation, San Francisco, Ca. (1986).

[6] Cassinis, R.: "An Example of a Distributed Intelligence Discrete Process Controller", Digital Systems For Industrial Automation, vol. 1, n. 1, New York (1981).

[7] Cassinis, R.: "Hierarchical Control of Integrated Manufacturing Systems", Proc. XIII International Symposium on Industrial Robots, Chicago (1983).

[8] Cassinis, R.: "BARCS: A New Way of Building Robots", Internal Report 87-036, Dipartimento di Elettronica, Politecnico di Milano, (1987).

[9] Chatila, R.: "Mobile Robot Navigation: Space Modeling and Decisional Processes", Proc. 1986 IEEE International Conference on Robotics and Automation, San Francisco (1986).

[10] Fox, B. R., and Kempf, K., G.: "A Representation for Opportunistic Scheduling", Proc. 3rd International Symposium on Robotic Research", Faugeras & Giralt, Eds., MIT Press, Cambridge, Ma. (1986).

[11] Gallerini, R., and Somalvico, M.: "General Structure of the Error Recovery Subsystem: Skeleton Extractor, Signal to Symbol Transformer, Sensory Robot and World Perceptor, Error Recovery Module", ESPRIT P623 Working Paper IP - PdM - 12.86/1, Milano (1986).

[12] Harmon, S.Y., Aviles, W.A., and Gage, D.W.: "A Technique for Coordinating Autonomous Robots", Proc. 1986 IEEE International Conference on Robotics and Automation, San Francisco (1986).

[13] Linden, T.A., Marsh, J.P., and Dove, D.L.: "Architecture and Early Experience with Planning for the ALV", Proc. 1986 IEEE International Conference on Robotics and Automation, San Francisco (1986).

[14] Paul, R.P., Durrant-White, H.F., and Mintz, M.: "A Robust, Distributed Sensor and Actuation Robot Control System", Proc. III International Symposium of Robotics Research, MIT Press, Boston (1986).

[15] Saridis, G.N., and Valavanis, K.P.: "Mathematical Formulation of the Organization Level of an Intelligent Machine", Proc. 1986 IEEE International Conference on Robotics and Automation, San Francisco (1986).

[16] Stenerson, R.O.: "Integrating AI into the Avionics Engineering Environment", IEEE Computer, (February, 1986).