ABC^2: AN ARCHITECTURE FOR INTELLIGENT AUTONOMOUS SYSTEMS

Vicente Matellán Daniel Borrajo

e-mail: {vmo, dborrajo}@ia.uc3m.es
Fax: + 34 1 624 94 30
Departamento de Informática
Universidad Carlos III de Madrid
C/ Butarque 15. 28911 Leganés. (España)

Abstract: This paper presents an architecture for the control of autonomous systems, that allows cooperation among them. This paper focuses on the description of the upper layers of the model. The control structure is based on a general purpose multi-agent architecture based on a two levels approach. One level is composed of reactive skills capable of achieving simple actions by their own. The other one is an agenda used as an opportunistic planning mechanism to compound, activate and coordinate the basic skills. This agenda handles actions both from the internal goals of the robot or from other robots.

Keywords: Robots, Agents, Co-operation, Architecture, Fuzzy.

1. INTRODUCTION

The field of intelligent autonomous robotic systems has strongly emerged after several years from the first attempts, such as the Shakey robot (Nilsson, 1984). These first systems used a classical AI view of planning and problem solving, which had, among other problems, the reasoning and acting on highly dynamic environments on real time. Then, research efforts as the one by Brooks (Brooks, 1986) focused the behavior of autonomous systems on using pre-defined "ad-hoc" behaviors (Connell, 1990). These architectures have shown to be very effective on some domains and on specific tasks. However, more flexibility is strongly needed when designing a general architecture that requires high-level reasoning working with symbolic information and high-level goals. This is the case of most real-world problems that an autonomous intelligent system faces.

This paper presents a multi-agent architecture, named ABC^2 (**A**genda **B**ased **C**ooperation for **A**gent's **B**ehaviors **C**oordination), based on pre-defined *skills* that each agent composes in an opportunistic way to achieve an intelligent behavior. An agenda has been used to keep a list of pending actions, where each action can require (or not) pre-defined simpler actions. Actions can be inserted into the agenda by other actions, by events from the environment or by requests received from other robots. Similarly, actions can be accomplished as a result of the execution of other actions, by another robot actions or simply by changes in the world. For the definition of the skills, different types of controllers can be used, such as fuzzy controllers (Matellán *et al.*, 1995*b*), mathematical calculus (Brooks, 1986), or learned behaviors (Stone and Veloso, 1995). In the case of ABC^2 ,previously developed software for building fuzzy behaviors will be used.

This general architecture supports the cooperation of heterogenous and autonomous systems.

This work has been influenced by different DAI (Distributed Artificial Intelligence) works (Bond and Gasser, 1988), particularly by the Coopera architecture (Sommaruga and Catenazzi, 1996)

and, in general, AI techniques, such as high-level agenda-based planning (Currie and Tate, 1991). This contribution extends this kind of systems in two aspects. First, these ideas will be applied to cope both with *dynamic* environments and with a *real world* environment (in the sense of a sensors and actuators approach). Second, controllers will be able to be designed using any reasoning paradigm. In particular, fuzzy logic (Zadeh, 1973) has been used, both to define basic controllers, and to write the heuristics that will control the agents.

In the next section, the architecture is presented in more depth, discussing the skills that are used. Section 3 describes an example of the execution of the whole system, specially the role of the agenda in the control of the robot actions. In the last section some conclusions and future work are presented.

2. DESCRIPTION OF THE ARCHITECTURE

An intelligent system, in particular an intelligent autonomous robot, will be defined as a knowledge structure defined by a set of statical and dynamic attributes. Among the static ones there is the name of the agent (N), the list of its skills (S), the knowledge about its team-mates names and skills, called yellow-pages (Y), the current state of the world, defined using a language (L), and the set of heuristic rules that governs the behavior of the agent (H). So, an agent (A) can be represented as the tuple: $A =< N, S, Y, L, H >$. In the same way, the team of agents can be represented as $< N, S, Y, L, H >^{+}$, given that a team is made up of at least one agent.

Among the dynamic information that defines the current situation of an agent there is the agenda (A_g) that contains the acts currently under consideration, the queues of messages (Q) received or pending to be sent, and the information (I) about the current state of the world, defined using the language L. So, an agent in a given moment is defined by $< A, A_g, I, Q >$, and the situation of the whole team as $< A, A_g, I, Q >^{+}$. This architecture is graphically shown in Figure 1. Its different parts are:

Skills Set of simple and reactive controllers. These controllers implement pre-defined behaviors that an individual system can accomplish. They can be implemented using any type of decision-making mechanism. In particular, for the example described in Section 3, fuzzy controllers (Zimmermann, 1990) have been used. The main reason for this election was the possibility of using previously developed and tested fuzzy reasoning libraries.

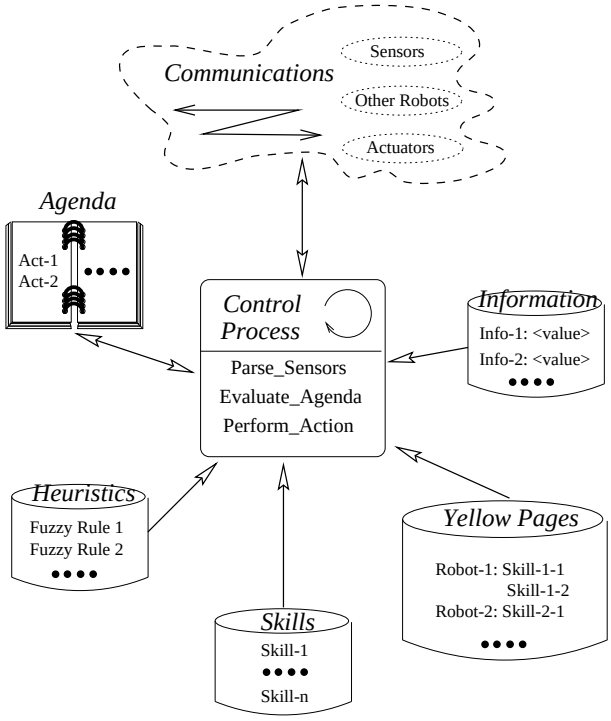

Fig. 1. Architecture of the Robots.

The design of the behaviors has been done heuristically. This means that the rules have been chosen by hand. However, many "automatic" methods for designing this type of robot behaviors can be found in the literature, ranging from the mathematical methods (Steels, 1990) to neural networks (Maes and Brooks, 1990) or genetic algorithms (Koza, 1991). For instance, good results have been obtained in previous works using the last method (Matellán *et al.*, 1995*a*). However, most of these methods have been designed to learn in well-defined environments, with few dynamic objects, and they are highly time consuming.

Yellow Pages Knowledge that an agent has about the other agents that form its team. This information basically consists of a table made by the name of its team mates, and the name of the skills they can accomplish. These skills will be used in the same way as its own skills.

A skill can be considered as an abstraction of an action that will be accessible to other team-mates. In fact, this means that the robot has meta-knowledge about itself (through its skills definition) and its team-mates (using the yellow pages).

Information Classical reactive behaviors compute the outputs for the actuators of an agent directly from the raw numerical data perceived by its sensors. In other environments, like for instance the RoboCup simulator (Kitano *et al.*, 1995), the inputs are not numerical data obtained from the sensors, but a mixture of linguistic and numerical information. In order to be able to handle this information, a reduced

high-level language is used. It allows the agent to define the inputs of the skills and to keep significant information about the current state of the world. So, the skills use this language to represent the information of the robot inputs.

Communication One of the distinctive capabilities of agents is their ability to communicate with other agents. In order to be able to handle the intrinsic complexity of the communication (protocols, queues, etc.) the agents are given a specialized entity to cope with it.

The Agenda The *Agenda* is a dynamic structure that contains items named *acts*, based on the Speech Acts theories (Cohen and Perrault, 1986). These acts represent the potential actions that the robot is considering at a moment. Four types of acts have been considered:

- `DO <skill>`, that represent potential skills that the robot can perform by itself. In the next section, the fundamentals of these behaviors are presented.
- `REQUEST <agent, skill>`, to ask another agent to perform a particular action.
- `REQUESTED <skill,agent>`, to indicate that the action in the argument of the act has been requested by another robot in order to be performed by this one.
- `SUPPLY_INFO <agent, info>`, to point out that some information has to be sent to another robot.
- `INFORMED <info, agent>`, to get a piece of information sent by other robot.

Heuristics Heuristics decide at any time what act to select from the agenda. Fuzzy rules have been used in the current implementation, although other types of heuristic representations are being considered. The input variables of these rules are, among others: the priority of the skill associated to an act; the time that an act has been in the agenda; the number of acts that require an act to be evaluated; the information from the environment; and the type of agent.

The output is the weight of each act in the agenda. Once the acts have been weighted, the eligible act to be executed is the one with the highest weight. These heuristics can also be used to purge the agenda of undesired acts.

3. SYSTEM EXECUTION

The definition of a particular skill (as shown on Figure 2) consists first on the design and implementation of the controller that performs the desired action (this is represented as the function *Execute* in the figure). Then, a condition for triggering the controller (named *Ready* in the figure) is established in order to know if the controller can be executed. In the case of the skill being evaluated but not being able to execute its associated controller (the *Ready* function returns a `FALSE` value), the skill provides a list of skills that can make it "executable". This list has been named *Needs* in the figure. The remaining slot is the *Priority* assigned to the behavior. This value can be used in the heuristic rules to select acts from the agenda.

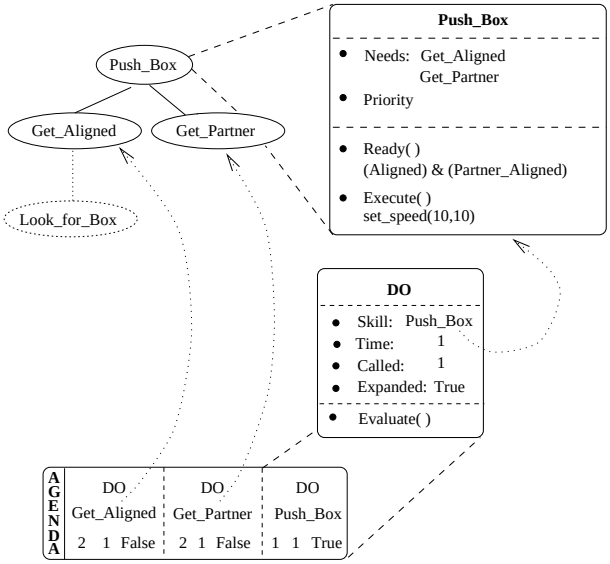

Fig. 2. Relations among the skills and the agenda in a simple two robotic agents domain.

Let us consider an example where two robots have to push a box at the same time from the same side. In order to simplify the problem, two points where the robots have to align to make an adequate pushing have been defined. These points correspond with two lights placed inside the box to push in the real environment of Figure 3.

A skill makes the robot push when it is aligned (`Push_Box`). Another forces it to be aligned (`Get_Aligned`). A third skill looks for the alignment point (`Look_for_Box`), and a fourth one asks the other robot for aligning (`Get_Partner`). Once the skills of the robots have been designed (in this example only the four skills that appear in the top-left tree of Figure 2 have been considered), the heuristics have to be defined. Let us suppose that a simple heuristic is settle up: "Select from the agenda the act whose *Priority* value is the highest from the ones that are *Ready*". Let us also suppose that the information that the robot has about the world is the raw data received by its sensors and the information about whether its teammate is aligned or not.

In order to achieve the task of having the robots push the box, they should be initialized. This is performed by inserting the act `[DO: Push_Box]` into the agenda. The components of an act (as shown in Figure 2) are: the type of the act (`DO`, `REQUESTED`, etc.); the name of the associated skill; the counters `Called`, that indicate the number of

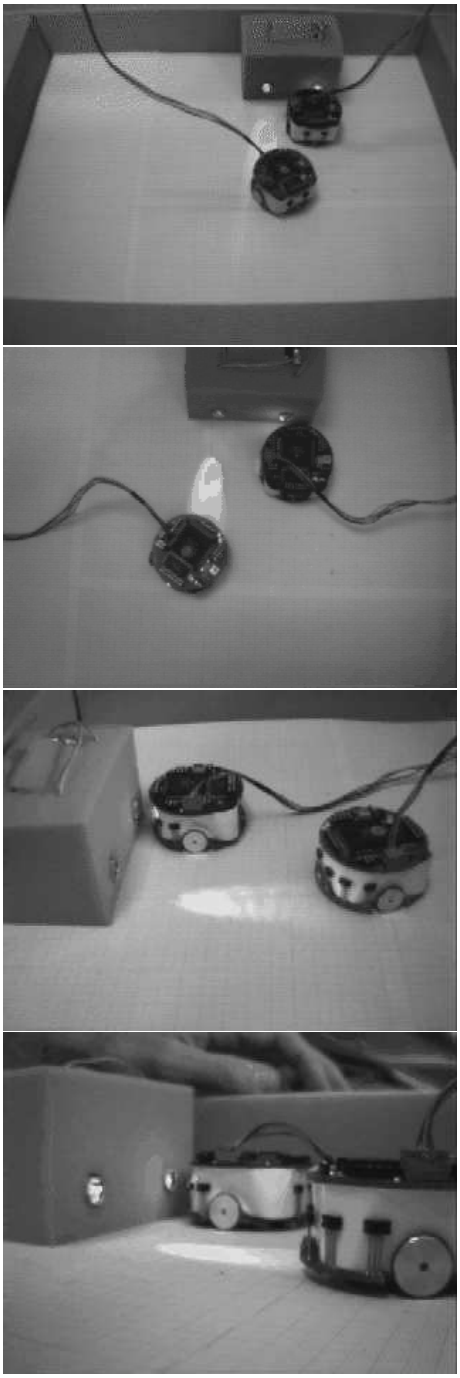

Fig. 3. The real environment.

acts that require it, and `Time` that keeps the time when the act was inserted into the agenda; the switch `Expanded`, that indicates if the needs of the associated skill have been added to the agenda or not; and the function `Evaluate`, that indicates what has to be done when the act is selected (for example, execute its associated skill if the type of the act is `DO`).

The way the control cycle works is as follows: first, the applicable acts are selected. This is achieved by consulting the *Ready* function of the skill associated to each `DO` act. If the act is not applicable, then the `Expanded` switch is checked. If it has not been expanded, its needs are inserted into the agenda as `[DO: <need>]` acts. This is the situation reflected in Figure 2 where the act `[DO: Push_Box]` was not applicable and it has been expanded by inserting the acts `[DO: Get_Aligned]` and `[DO: Get_Partner]` into the agenda.

The addition of needs checks if that act had been previously added to the agenda by other acts. If the act already was in the agenda, the counter `Called` of the act is increased; otherwise, a new act is added to the agenda. On the other hand, if the act is applicable and had been expanded, the counter `Called` is decreased. At the same time that the applicable acts are selected, the acts whose `Called` counter is equal to zero (no other act requires them) are removed from the agenda. Once the applicable acts have been selected, the domain heuristics are applied to select the one that will be evaluated.

Another way acts can be inserted into the agenda, apart from their insertion as needs of other acts, is directly by the `Execute()` function of a skill. For instance, the execution of the skill `Get_Partner` may result in the insertion in the agenda of an act such as `[REQUEST: RobotB, Get_Aligned]`. The information of the skills of other robots is in the Yellow Pages. The evaluation of this act would result in sending this request to the other robot.

The treatment of the other types of acts is similar. Only the `Evaluate()` method (see Figure 1) of these acts is different. For instance, if the act `[REQUESTED: Get_Aligned, RobotA]` is evaluated by the RobotB, it would produce[1] the insertion of the act `[DO:Get_Aligned]` in its agenda. In a similar way, the evaluation of the skill `Get_Aligned` would cause the insertion of a `[SUPPLY_INFO: RobotA, Aligned]` act into the agenda. The evaluation of this one will generate an act `[INFORMED: Aligned, RobotB]` into the agenda of the first robot, and its evaluation in toggling the predicate `Partner_Aligned`.

4. CONCLUSIONS AND FURTHER WORKS

In this paper the ABC^2 architecture has been presented. Its fundamentals have been explained and also the theoretical principles that have influenced it. Then, an example of operation has been presented in order to show how the system works.

The practical experiments have shown that this architecture is well suited to domains where there is no need of great flexibility in the accomplishment of the skills, that is, environments where

[1] Or not, if the heuristics of the second robot decide, for example, that acts containing requests from RobotA are discarded.

opportunistic planning can be used. Besides, it allows an intuitive method to deal with cooperation among agents by letting agents define their own skills, and the rest of the group having knowledge of them.

Currently, ABC^2 is being used on the robo-soccer (Matellán and Borrajo, 1997) domain in order to test its abilities when facing highly dynamic environments. This experiment will let us probe whether ABC^2 is flexible and general enough to face different problems or not.

5. REFERENCES

Bond, Alan H. and Les Gasser (1988). *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann.

Brooks, Rodney A. (1986). A roboust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation* **RA-2**(1), 14–23.

Cohen, Philip R. and C. Raymond Perrault (1986). Elements of a plan-based theory of speech acts.. *Cognitive Science* **RA-2**(3), 177–212.

Connell, Jonathan H. (1990). *Minimalist Mobile Robotics: A Colony-style Architecture fo a Mobile Robot*. Academic Press. Cambridge, MA. Latas de Coca-Cola.

Currie, Ken and Austin Tate (1991). O-Plan: the open planning architecture. *Artificial Intelligence* **52**(1), 49–86.

Kitano, Hiroaki, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda and Eiichi. Osawa (1995). Robocup: The robot world cup initiative. In: *Proceedings of the IJCAI-95 Workshop on Entertainment and AI/Life*. pp. 19–24.

Koza, John R. (1991). Evolving emergent wall following robotic behavior using the genetic programming paradigm. In: *Toward a practice of autonomus systems. Proceedings of the First European Conference on Artificial Life* (F.J. Varela and P. Bourgine, Eds.). MIT Press and Bradford Books. Cambridge, MA. pp. 110–119.

Maes, Pat and Rodney Brooks (1990). Learning to coordinate behaviors. In: *Proccedings of the Eighth National Conference on Arfificial Intelligence*. Morgan Kaufmann. San Mateo, CA. pp. 796–802.

Matellán, Vicente and Daniel Borrajo (1997). An agenda-based multi-agent architecture. In: *Proceedings of the First International Workshop on RoboCup in conjunction with RoboCup-97 at IJCAI-97* (Hiroaki Kitano, Ed.). Nagoya (Japan).

Matellán, Vicente, José M. Molina, Javier Sanz and Camino Fernández (1995*a*). Learning fuzzy reactive behaviours for autonomous robots. In: *Proceedings of the 4rd European Workshop on Learning Robots*. Karlsruhe, Germany.

Matellán, Vicente, José Manuel Molina and Camino Fernández (1995*b*). Fusion of fuzzy behaviors for autonomous robots. In: *Proceedings of the Third International Symposium on Intelligent Robotic Systems*. Pisa, Italia.

Nilsson, Nils J. (1984). Shakey the robot. Technical report. SRI A.I.

Sommaruga, Lorenzo and Nadia Catenazzi (1996). From practice to theory in designing autonomous agents. In: *First Australian Workshop on Distributed Artificial Intelligence*. Vol. LNAI-1087 of *Lectures Notes in Artificial Intelligence*. pp. 130–143. Springer-Verlag.

Steels, Luc (1990). Exploiting analogical representations. In: *Designing autonomous agents: theory and practice from biology to engineering and back* (P. Maes, Ed.). pp. 71–88. MIT Press and Bradford Books. Cambridge, MA.

Stone, Peter and Manuela M. Veloso (1995). Beating a defender in robotic soccer: Memory-based learning of a continuous function. Technical Report CMU-CS-95-222. Computer Science Department, Carnegie Mellon University.

Zadeh, Lotfi A. (1973). Outline of a new approach to the analysis of complex systems and decision processes. *Transactions on Systems, Man and Cybernetics*.

Zimmermann, Hans-Jurgen (1990). *Fuzzy Sets. Theory and its Application*. Kluwer Academic Publishers. Boston, MA (USA).