

ROBOT SKILL ABSTRACTION FOR AD ARCHITECTURE ¹

R. Rivas ^{*,2} A. Corrales ^{*,2} R. Barber ^{*,2}
M. A. Salichs ^{*,2}

** Carlos III University of Madrid - Spain*

Abstract: In this paper a set of tools is presented. This set of tools allows obtaining an approach to the development of an architecture for personal robots control which is based on robot skills, specifically for the AD Architecture developed in the Robotics Lab research group at the University Carlos III in Madrid. The paper describes three different software components: an event emission and capture system, an information distribution system and a skill template class. A brief description of the AD architecture is done in order to understand how the components must work. Then the three tools that are used are explained in detail. A description of our robot, Maggie, in which the tools have been combined, is done. Finally, the development stage and the experimental results are shown.

Keywords: Personal robots, software architecture, skills.

1. INTRODUCTION

In the last few years the robotics field has been evolving, trying to be more and more useful to mankind and trying to integrate to its environment. The interaction between man and robots would be greater when robots look more like human beings, not only physically but also in the psychic aspect (Furuta *et al.*, 2001; Dario *et al.*, 2001). In the psychic aspect, the way of reasoning, navigating and learning is copied. Also, attempts have been made to reproduce the human emotional structure and complete architectures are even designed that try to implement the

mental levels and emotional abstraction of human beings.

One of the first robot control system definitions is the one proposed by Walter Jacobs in (Jacobs, 1972); the robot behavior depends on the chosen *strategy*, this strategy is emitted by a system component named *control*. Maja Mataric offers in (Mataric, 1992a) a more extensive definition, where it states that a robot architecture is mainly a way to organize a control system and, additionally, the architecture imposes restrictions to the way in which the control problem must be solved. Another definition of robot architecture is the one provided by Ronald Arkin in (Arkin, 1998, p. 124), where it states that robot architecture is “the programmed specifications and systems that provide the tools and languages for the creation of systems based on behaviors”.

Many different techniques and approaches for robotic control have been developed. Two basic approaches exist that have emerged as a result from this development: interest in deliberative

¹ This work was supported by R2H Project, funded by the spanish Ministry of Science and Education, and Robocity2030 Project, funded by the Madrid Community Spain.

² R. Rivas, A. Corrales, R. Barber and M. Salichs are with Department of Systems Engineering and Automation, Carlos III University, Madrid, Spain. R. Rivas is too with Los Andes University Merida - Venezuela. (rafael.rivas, anavalle.corrales, ramon.barber, miguel.salichs)@uc3m.es

processes taken as examples (Albus *et al.*, 1989) developed by James Albus and interest in the reactive processes whose main feature was presented in 1986 by Rodney Brooks in (Brooks, 1986).

It was only a matter of time before researchers began to test hybrid systems that joined some of the characteristics of the two approaches. Among the first ones was Maja Matarić (Matarić, 1992b), in its proposal a planning layer based on multiple reactive layers is added.

AD Architecture is an hybrid architecture that tries to include deliberative and automatic concepts in a two layer architecture based on skills using events and share memory to communicate among them. At the present, an emotional engine is being implemented in order to proportionate the robot human like behavior.

In section 2 the AD Architecture, in which this work is included, is going to be presented. In section 3 the main contribution of this work is emphasized. In section 4 and 5 the way in which the event system and shared memory has been developed are described. In section 6 the idea of how the skill is abstracted is exposed. Finally, section 7 shows an example of how skills work in a collaborative task, using the proposed mechanisms.

2. AD ARCHITECTURE

In the human being two mental activity levels can be differentiated: the Deliberative level and the Automatic level. Being based on these ideas, in the AD architecture only two levels are present: Deliberative and Automatic (Salichs and Barber, 2001). The Deliberative level is related to reflective processes and the Automatic level to automatic processes.

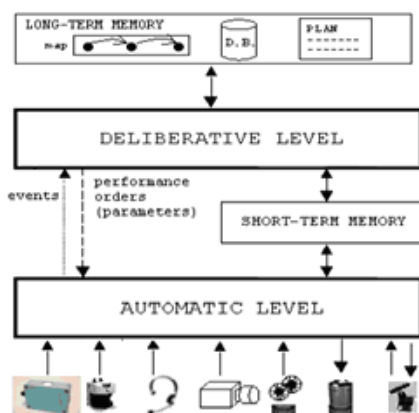


Fig. 1. Layers in the A/D architecture.

Both levels have a common characteristic: they are constituted by **skills**. The skills are the different capacities to reason or to carry out an action.

These skills are made possible by the running command produced by other skills or a sequencer, giving back the data and events to sequencer or other skills. Those skills are the base of the AD architecture. Figure 1 shows the representation diagram of the AD architecture.

Communication between the Deliberative level and the Automatic level is bidirectional. Both levels communicate through shared memory.

2.1 Deliberative level

In the Deliberative level are located the modules that require reasoning or decision capacities. These modules do not produce immediate answers and need some time to process the information with which they work to make decisions.

Therefore, the Deliberative level of this AD architecture is formed by the deliberative skills and the main sequencer, as it is represented in Fig. 2. This sequence is established beforehand and it is going to determine the robot behavior and its way of reacting to different situations.

2.1.1. Deliberative skills The deliberative skills are each one of the capacities of reasoning and decision-making on which the autonomous system counts on. They are the ones in charge to handle and to modify the content of long-term memory, as well as to handle and to execute the automatic skills. As it was previously stated, the deliberative skills are handled and controlled by the Main Sequencer. The Deliberative level skills are also able to generate events that they communicate to the Main Sequencer, so that it decides the new deliberative skills to enable. Examples of these deliberative skills implementation are the planners, the navigators and the modules that carry out the environment modelling.

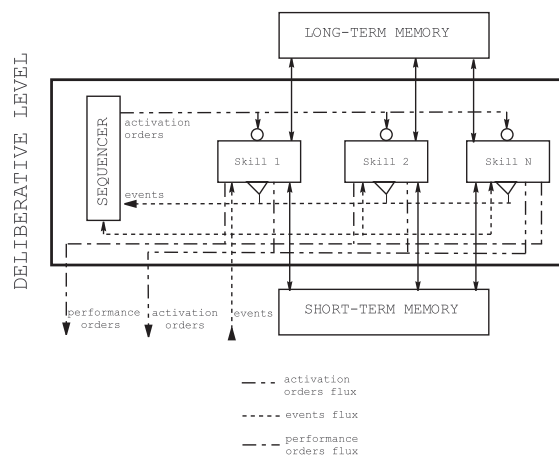


Fig. 2. AD architecture: Deliberative level.

2.2 Automatic level

The Automatic level is in charge of the control, at low level, of the devices of the robot. This level allows the robot to have the necessary reaction capacity to quickly respond to environment changes (Salichs and Barber, 2001; Salichs *et al.*, 2002). Figure 3 shows the elements that make up this level: Virtual sensors and actuators, reflex actions and automatic skills.

The Automatic level must fulfil the following requirements:

- The elements that make up this level must be able to run in parallel and to interact with others.
- It must allow adding new elements without affecting the performance of other elements, so that the architecture is expandable and dynamic.
- It must allow the generation of complex skills from the existing ones. A skill can be used by several complex skills, which gives flexibility to the architecture.

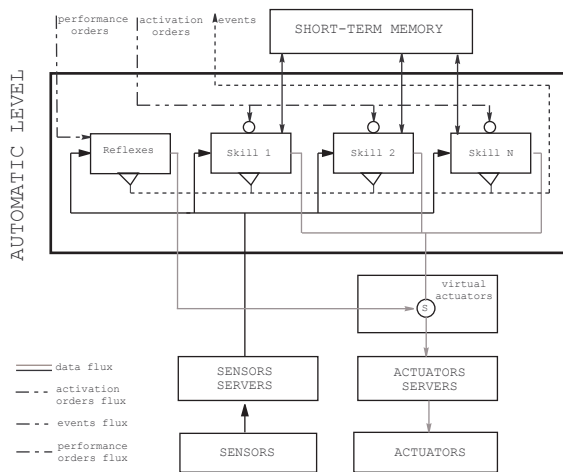


Fig. 3. AD architecture: Automatic level.

2.2.1. Automatic skills The automatic skills are defined as the capacity to process information from the virtual sensors and/or to generate running commands on the virtual actuators. All the automatic skills have the following characteristics:

- They can be enabled by skills located in the same level or in the Deliberative level. A skill can only disable skills that it has previously enabled.
- They can share the information generated by others skills.
- They can generate several events and they will notify them only to those skills that have previously requested them.

The simple automatic skills of the AD architecture are the basis for the complex skills generation.

These skills can be combined as well to generate more complex skills. The skills modular feature allows easily using them to create skills hierarchies with greater level of abstraction. The automatic skills are not planned beforehand. The skills are dynamically configured based on the task the robot must carry out and on the environment conditions. With this a greater flexibility in robot behavior is obtained.

2.3 Types of AD architecture memories

The AD Architecture is made up of two types of memory: short term memory and long term memory.

- The Short Term Memory stores the most important data from the sensors and shared data among skills.
- In the Long Term Memory the permanent knowledge is stored. The robot uses this knowledge to reason or to make decisions.

3. MAIN CONTRIBUTIONS TO AD ARCHITECTURE DEVELOPMENT

The proposed developments of this work are based on building a control system based on distributed and uncouple components. This proportionates several tools to the research:

- The Abstract Class “Skill” describes the global behavior of a skill. It defines an empty method called “process”. This method describes the control loop of the skill. The class constructor is designed in order to activate one or more times the control loop. It includes a method in order to activate or deactivate the control loop from an external class. All the multiprocessing details are hidden to the user.
- The Event Manager allows the event transmission from running process in the same computer or from computers connected via TCP/IP protocol. This fact allows the implementation of extended status machines with a higher abstraction level. A skill that needs to use a event manager can subscribe to an event and define a particular behavior in the same instant in which the event occurs. All details related to connection establishment, send / receive and close connections as well as the threads programming of the control process are hidden to the user.
- The Shared Memory System allows to create memory areas shared by all process. Process can be allocated in different computers and data is distributed in a serial way. All connection process, memory management and data transmission are hidden to the user.

These three components have been designed, developed and tested in the Automatic-Deliberative (AD) control architecture frame, proportionating a modular and easy way of integrating and debugging skills and communications in the architecture. All mechanism of threads management, processes and communications are transparent to the user. Currently, there are some tools available for the users, but unfortunately their code is closed to the developers and hardware or libraries dependent, therefore, hidden to programmers. The tools presented in this paper can be used in group or individually and they can also be extended to different platforms (Windows, Linux) and different programming languages like C++, Java Python.

4. EVENTS EMISSION AND CAPTURE SYSTEMS

In AD Architecture skills behave in a cooperative way and, in some cases, one or several skills are subordinated to other skills. The ideal mechanism for the skills behavior coordination in AD architecture is carried out by means of the events emission and capture. To obtain a successful performance it is necessary to have a mechanism of event management, but this management must be carried out having in mind that skills are basic components and in most cases reusable for different applications.

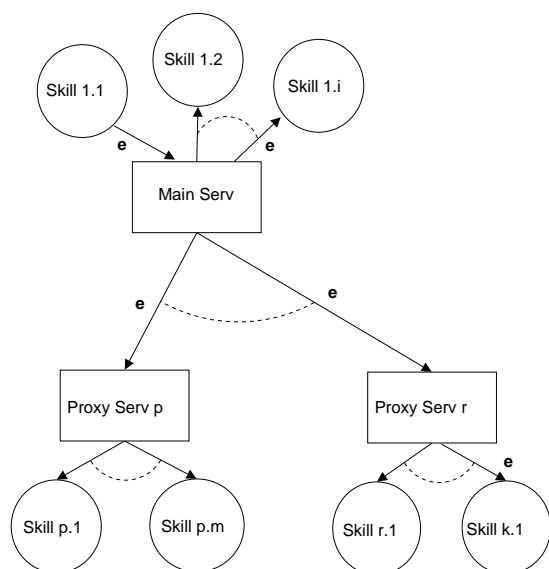


Fig. 4. Event spreading.

The managing system of events used in our robots is loosely coupled, a skill that generates events does not know beforehand whether or not other skills exist that will be willing receive the emitted event. These allow having skills devoted to a specific task, hiding the communication mechanisms such as the events addressing and the transmission to receivers. For example, when a skill designed

to detect obstacles finds one of them, it emits the “*obstacle dected*” event. Those events can have parameters specifying the nature and location of the obstacle. The skill does not know if the emitted event was captured and processed by the skills in charge of capturing the event.

The design of the managing system of events is accomplished by the implementation of the publisher/subscriber design pattern described in (Gamma *et al.*, 1997). In order to allow the events distribution between enabling skills in different computers, a main server and several secondary servers were created. The performance of the event management system is shown in Fig. 4. An event is generated by a skill and given to the event manager by the method *emit*. The local event manager places it in the mailboxes of each one of the skills subscribed to that event, and the distributed component of the events manager spreads it towards the secondary servers (proxies), and these secondary servers place it in the skills mailboxes subscribed to that event.

If a skill needs the use of events, it instances an object of the class *CEventManager* in its local memory space. *Subscribe* and *Emit* are the main methods of this class.

5. INFORMATION DISTRIBUTION SYSTEM

Sharing data between skills is a basic feature in AD architecture; from the information processed by the sensors to the information used in the deliberative layer it must be available for current or future skills. In some cases, like in the physical sensors, specific skills exist that process the obtained data and that extract information to distribute it among the skills. When creating the AD architecture the data flows obtained from the physical sensors are extracted in a synchronous way with transmission periods that go from 50 msec in the sonar sensors to 120 msec in the laser. But the information obtained from the data is asynchronous. For example, if a door is found, the laser begins to obtain data and to process it, when the conclusion is reached that a door is been detected the door coordinates are available for other skills and an event is emitted. The information to be distributed is usually a summary of the data obtained from the sensors, and in some cases it can be transmitted by the events management system as parameters.

Not only data from the physical sensors to skills in the automatic layers is transmitted, but also data are sent among skills in the deliberative layer to the automatic layer, and among skills in the same layer.

Additionally to the events management system, a system of distributed shared memory was created. This system is based on the following features:

- The system must be able to distribute different data types.
- Additionally to the data, the date of the data capture must be stored.
- The data must be available to all the skills of the control architecture.

The created system has two components, a simple component based only on POSIX for the local distribution and a distributed component managed through servers with *well known* addresses. They are characterized by:

- Data are registered and classified as semantic types, although syntactically they have the same structure; and they are identified by a unique key.
- The data are stored as bytes flows in servers hierarchy similar to the ones used in the events management system.
- The current and the previous values are stored. When writing a new data, the previous data is not eliminated, this is stored like the previous version, the behavior is similar to a stack, and in this case of depth 1.
- Several skills can read a same data in a concurrent way.
- If a skill needs to use the shared memory, it must create an instance of the class used as interface to the system. The methods of this class allow to register and to eliminate data structures, as well as the reading and writing of a particular data.

6. SKILL ABSTRACTION BY OBJECT CONSIDERATION

Some skills have been defined within the AD architecture (Rivas *et al.*, 2006), but since it is a recent creation, most of them have yet to be developed, and others are only in the planning stage. This fact entails the need to establish a skill design pattern; the result is a generic class (Template Class). It is of course an abstraction to describe the main skill aspects without specifying the details.

An instance of a Skill is characterized by:

- Having three states: Ready, running and blocked.
- Having three running ways: cyclical, periodical and by events.
- The organization:
 - Each skill is a process, the communication between processes is by events and/or shared data.

- A skill is represented by one or more tasks or by adding skills.

- All the handling of multiprogramming, communication between processes, composition, etc., must be transparent to the user, or at least simple to use.

The created component has the synchronization services, creation of sub processes (threads) and timing. The user chooses the way of integration and organization among them. The control loop must be defined depending on the task of the skill. Combining or extending skills it is possible to obtain new improved skills.

7. MAGGIE: EXPERIMENTAL PLATFORM



Fig. 5. Robot Maggie

From hardware point of view, Maggie (Salichs *et al.*, 2006) (RoboticsLabs UC3M, 2006) was initially builded from a Magellan Pro robot distributed by iRobots. Later, this base was replaced by a homemade platform. Nowadays, Maggie has a design of a 1.35 meters tall girl-like doll. The base is motorized by two differentially actuated wheels and two caster wheels. It is also equipped with 12 bumpers, 12 infrared optical sensors and 12 sonar sensors, a laser range finder (Sick LMS 200) and several tactile sensors in the upper part of the robot. On top of the platform, an anthropomorphic robot head with an attractive, well-groomed appearance has been added. The head includes vision and speech capabilities (Fig.5). Inside Maggie, there are three computers interconnected through an on-board Ethernet and connected with external computers through WiFi 802.11.

8. EXPERIMENTAL RESULTS AND CONCLUSIONS

The creation of independent and disconnected components allows the integration and the crea-

tion of systems programmed in a fast and comfortable way, moreover to facilitate the maintenance. The components can be modified and be replaced by new versions without spreading the changes to other components as long as the minimum interface stays invariable. Thanks to the vision of the concept of generic skills, it is possible to create a simple control system from proven components. These ideas have been applied in the next approach, the three components have well defined goals and are easily integrated.

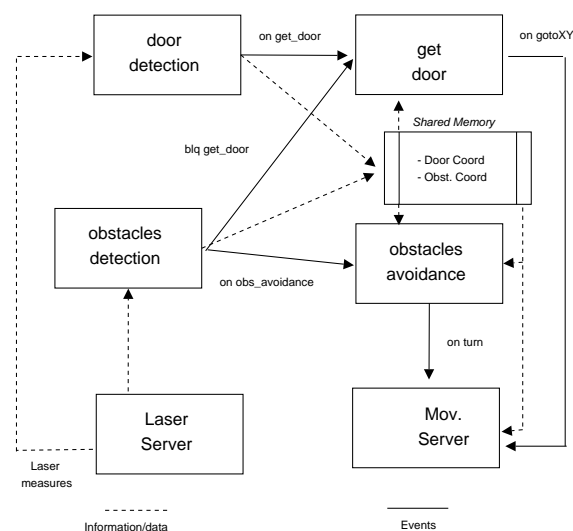


Fig. 6. Cooperative work among components.

In Fig. 6 an application that shows the skills cooperative work, the events manager and the shared memory is presented. The aim of this robot task is to wander, looking for a door.

In this case the *getDoor* skill is subordinated to the *obstaclesDetection* skill. When the robot detects a door its coordinates are placed in the shared memory and the *getDoor* event is enabled. The *getDoor* skill is enabled planning the actions to move to the door, placing the information that is going to be needed in the shared memory. Next, the event on *gotoXY* is enabled so that the movement server creates the commands for the robot actuators. When approaching the door, if an obstacle is detected, the *blq getDoor* event is enabled to place the *getDoor* skill in the blocked mode and enable an evasive movement in the robot too.

Other skills, not related to navigation tasks, as face detection or voice recognition has been developed using the proposed implementation.

The used mechanism has allowed to provide a skill development pattern in Maggie under the AD architecture, obtaining a greater benefit of the robot resources and a short period of development time. So far, the tasks planner under the AD architecture is being designed; when that stage is

accomplished, complete control architecture under the AD architecture must be created.

REFERENCES

- Albus, James S., John C. Fiala, Albert J. Waverling and Ronald Lumi (1989). Nasrem – the nasa/nbs standard reference model for telerobot control system architecture. In: *20th International Symposium on Industrial Robots*. Tokyo, Japan. pp. 4–6.
- Arkin, Ronald C. (1998). *Intelligent robots and autonomous agents*. MIT Press.
- Brooks, Rodney (1986). A robust layered control system for a mobile robot. *IEEE Transaction on Robotics and Automation* **2**(1), 14–23.
- Dario, P., E. Guglielmelli and C. Laschi (2001). Humanoids and personal robots: design and experiments. *Robotic Systems* **18**, 673–690.
- Furuta, T., T. Tawara, Y. Okumura and M. Shimizu (2001). Design and construction of a series of compact humanoid robots and development of biped walk control strategies. *Robotics and Autonomous Systems* **37**, 81–100.
- Gamma, Erich, Richard Helm, Raph Johnson and John Vlissides (1997). *Design Patterns*. Addison-Wesley Professional.
- Jacobs, Walter (1972). Control systems in robots. In: *ACM'72: Proceedings of the ACM annual conference*. ACM Press. New York, NY, USA. pp. 110–117.
- Matarić, Maja J. (1992a). Behavior-based systems: Main properties and implications. In: *IEEE International Conference on Robotics and Automation*. Nice, France. pp. 45–54.
- Matarić, Maja J. (1992b). Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation* **8**(3), 304–312.
- Rivas, R et al. (2006). Schab: Sistema minimo de supervision y control de habilidades en el robot maggie. In: *Jornadas de Automatica*. Almeria, Spain.
- RoboticsLabs UC3M (2006). Maggie web page. <http://roboticslab.uc3m.es/maggie>.
- Salichs, M. A. and R. Barber (2001). A new human based architecture for intelligent autonomous robots. In: *The Fourth IFAC Symposium on Intelligent Autonomous Vehicles*. Sapporo, Japan. pp. 85–90.
- Salichs, M. A. et al. (2006). Maggie: A robotic platform for human-robot social interaction. In: *IEEE International Conference on Robotics, Automation and Mechatronics (RAM 2006)*. Bangkok. Thailand.
- Salichs, M. A., R. Barber and M. J. Boada (2002). Visual approach skill for a mobile robot using learning and fusion of simple skills. *Robotics and Autonomous Systems* **38**(3–4), 157–170.