

# Dynamic programming for robot control in real-time: towards a morphology programming

**Mickaël Camus**

L.E.R.I.A., {Epitech.}  
24 rue Pasteur  
94270 Le Kremlin Bicêtre, France  
LIP6 UMR 7606 Paris VI  
UPMC 4 Place Jussieu  
75252 Paris Cedex

**Alain Cardon**

LIP6 UMR 7606 Paris VI  
UPMC 4 Place Jussieu  
75252 Paris Cedex

**Abstract** - Industrial or personal robots need more flexibility to manage a large-scale of contexts in instable environment. Currently, for each robot building, there is a conception, a design and a development to adapt the robot to an environment and a context. In this paper, we present a method based on a multiagent system to move towards a generic algorithm in order to control robot in real time. We present current problems for robots conception, features for the dynamic programming and technics and model to build the system. To finish, we expose different experiments based on the Aibo ERS7 by Sony, we observe the behavior of the robot according to its ontology and goals. Currently, a work on the synchronization between knowledge and action is in progress to move towards a more natural physical behavior.

**Keywords:** automated reasoning, behavior-based control, control, decision-making, multiagent systems.

## 1 Introduction

Lot of applications exist in the computer science world. All these applications touch a set of sectors in the market jobs. It is crucial for all company to update and to make evolves their softwares to increase flexibility and decrease costs. Currently, all development are specific to a need and to a hardware. For each evolution, there is a new model, a new conception and a new development. Lot of researchs are in progress to solve the genericity problem and the code generation.

We present a system to make evolved an algorithm in the case of dynamic evolution of inputs, outputs, or goals. First we start by situating the problem and give a research direction then we explain the modeling and the system conception. Finally we describe a set of experimentations.

## 2 Approach

In this work, an entity is described as a machine with a body and an artificial brain which is not embedded for performance reasons such as shown in the figure 1. This approach follows that of Damasio in [7]. Contrary to Descartes in [8], the body and the spirit are processed in synergy. It is with the body that the spirit can treat different information in an environment.

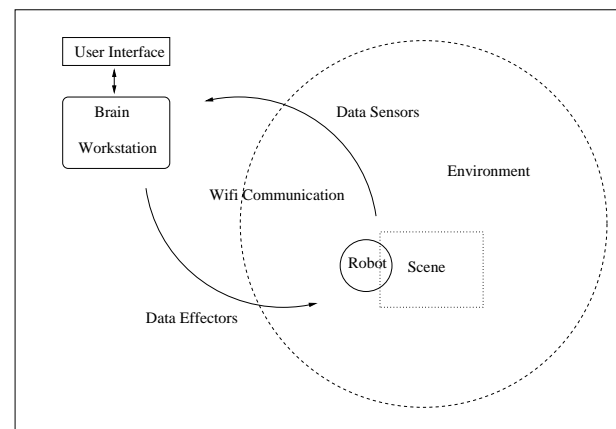


Figure 1: Brain workstation and its environment

The artificial brain is linked to sensitive sensors, effectors, position sensors, camera etc. All data is processed in parallel to superpose information and make an interpretation. We consider five processing levels for decision-making in an unstable environment as described in [4]:

1. Represent a contextual situation.
2. Direct the attention on particular elements (objects or actions).
3. Verify if these elements can be used to succeed a goal.
4. Build behavior action plans.

5. React to an object action feedback.

All these levels composed a systemic loop described by:

**sensors** → **representation** → **interpretation** →  
**action plan** → **effectors** → **sensors**.

Notice that the entity is in continue processing.

### 3 Problems and direction

In the case of a robotic development, there is an embedded software to give a minimal autonomy to the robot, so for each robot, we have a specific software. Moreover, if there is an evolution on sensors or effectors (add or delete for example), developpers have to modify the software, and it is true for all robots in the community in the case of multirobots management. We will suggest a new solution to manage dynamically all evolutions on a robots without new modeling and new modification on the software. We follow the Brooks approach presented in [2] and in [1]. However, the proposed solution can be processed on a personal computer with a basic configuration (1Ghz, 778 megaoctets of memory). We present a robotic case but it can be also used for any information problems such as autonomous spacecraft or Unmanned Air Vehicles to make evolved a static architecture such as presented by Guettier and Poncet in [6] towards an adaptative architecture presented by Cardon in [5].

All robots need a set of elements to be processed:

- Sensors lists.
- Effectors list.
- Knowledge.
- Goals list.
- Error managment.

We can consider these elements such as parameters. The aim is to change dynamically the algorithm to succeed a goal which can changed in timeline. We consider that our system is not embedded in the robot but is processed on a distant machine. There is a continued communication between the hardware and the artificial brain. Such as shown in figure 2, when we execute the system for the first time, the hardware sends its list of sensors and effectors to the distant machine. After this, there is continued communication to respect the systemic loop.

A sensor and an effector have a fixed architecture. It is a hardware with a set of allowed value or a flux such as a video flux. After the initialisation of all sensors and effectors, we can replace values by a string such as “caress” for the sensor on the head of the aibo or “ball” if the robot recognize a ball in the environment.

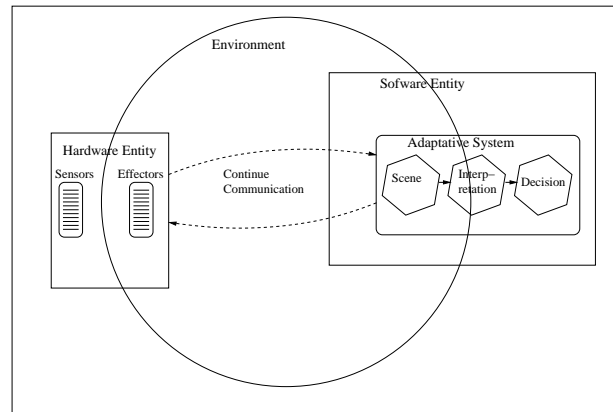


Figure 2: Communication between the hardware entity and the software entity. The hardware entity sends the list of sensors and effectors to the first connection. After this, there is continued communication to respect the systemic loop.

### 4 Features for dynamic programming

To implement an algorithm, we need different elements to store, classify and process data. In general, to create a program we have:

- Types: to know how to treat data (int, char, char\*, float ...).
- Variables: classify and store data.
- Instruction Functions: to direct the processing in the automat (if, while, for, switch ...).
- Operator and equality test (+, -, /, ==, ||, && ...).

For each program we have a limited set of variables with different types classified in different structures. All names of these variables are known by the developer. A set of these variables are linked with program parameters, so these variables are also known by the user. The software has a specific context, ie a knowledge field. Variables which are not linked with parameters are defined with the knowledge field, so a user can know these variables. At this point, the goal is to generate a good number of variables with their types to process the initialisation.

After the variables generation, we have to use variables to respect the algorithm, so to use operator, equality test and control functions which allow treat data according to their types. Here the goal is to find a solution to emulate each of these features.

According to the previous similitude presentation we can conclude that we need the following features to create a generic algorithm:

- Variables generation according to a knowledge field (application domain).

- Knowledge on variables types for the dynamically creation.
- Parameters to initialize variables.
- System aims to emulate operator, equality test and control functions.

## 5 Variables generation and instantiation

Each variable is associated to one or several functions with different parameters. All this information is in the ontology. This one describes all knowledge of the system with a description of the associated valid operation. The total description of the ontology is presented in another submitted paper. More details can be obtained from authors. We have a classification: “capacities” for all physical capacities, “peoples” for all known persons, “colors”, “object” etc, this list is not exhaustive, this classification evolves in the timeline. For example, in the class “object”, we can have: keyboard, ball, book, generator, cushion in the case of a personal robot such as Aibo by Sony. Each element of the ontology can be represents a simple knowledge or a variable with a specific type and value.

We parse all words (with type and value) in the ontology to build a multiagent system with these features (developed with the Oz/Mozart System presented by Van Roy in [9]):

- Message passing.
- Asynchronous communication.
- Thread management.
- Message control.
- Message delay.

Each knowledge in the ontology has a field, ie a minimum value and a maximum value. For a physical capacity, it is the limits of the engine, and for an object recognition, it is a rate (an object is recognized with a rate a 65 per cent for example.). For each knowledge in the ontology (we call this a “role”), the generated agent number depends to the difference between the maximum and the minimum. So, there are usually more agents then the knowledge in the ontology. For example, with the role “keyboard” we have a minimum rate of 70 and a maximum rate of 95, so we have fifteen agents for this role with a value between 70 and 95 for each agent. We have a representation of the system in a matrix view on the figure 3. With this section, we have the generation and the initialisation of the variables with their different types.

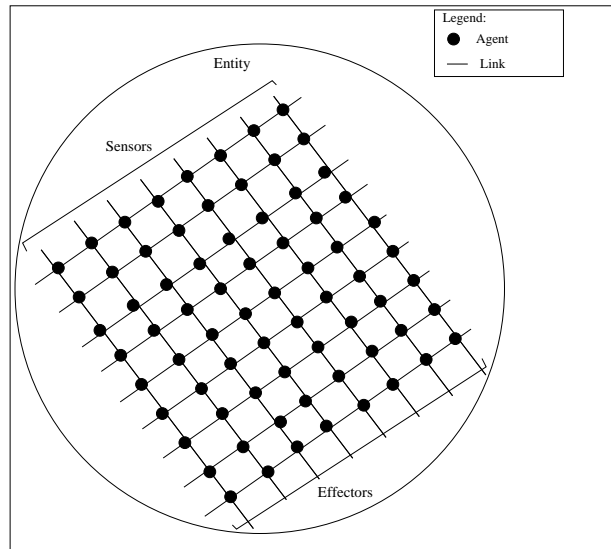


Figure 3: A two dimensional view of the agent matrix. Each agent manages one knowledge. The unit of knowledge forms a set of variable in the system. All the links between agents are not presented here. Each agent is linked with the others. To ensure a quick response in an unstable environment all the agents are mapped in the memory.

## 6 System processing

The software which manages the robot (a multiagent system) receives all sensors values in input to make a scene representation, interpret data, build an action plan and send effectors values to the output. All agents receive all value of all sensors. If a value matches an agent role, this agent is in activation with an exchange data with other agent which has a link with it (Links are acointance between agents. We explain this feature in the section Experiments and monitoring). The activation rate of a role give the value of the associated sensor in the input. We measure an agent activation with the message number send towards linked agents. If we observe the system such as shown in the figure 4, we see all active roles. With the processing, we have a method to emulate control instructions but we have to give a direction to the agents to respect the algorithm according to goals. We can directed the system with the morphology presented after this section.

## 7 Morphology to control

Control is a crucial section for the multiagent system, it is the core of the dynamic programming . All agent in the system are treated in parallel, so, when several agents activate itself simultaneously, a geometrical form is created. The phenomenom is called morphology, it has been discovered by Thom and presented in [10]. To control the multiagent system we rely on Campagne model presented in [3]. This model is an adaptation of Thom’s morphology [11] for multiagent systems. Notice that we have adapted this model for an asynchronous

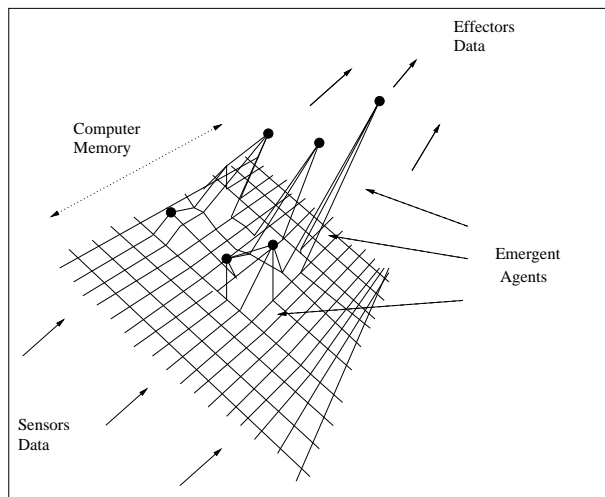


Figure 4: Here, we see an emergence of agents according to data coming from sensors. With this method, it is possible, at every moment, to have a description of the current scene, the composition of a context in an environment, ie when, where, what. All agents are mapped in memory.

communication.

Each system goal has a specific form, ie a specific value for each sensor linked to a goal. These values are included in a particular field and have to be in this field to respect the goal. We call this part the Natural Behavior Restriction (NBR). A goal has:

- A sensors list.
- A planning linked to the sensors list.
- A field for each sensor to know if a sensor has a good value during the processing: the NBR.

The aim is to direct the system behavior (communication between agents) towards the form linked to the goal (in the case of a simple goal). If the goal is more complex, there is a planning for all subgoals processing and a specific form per subgoal. A geometrical form is a specific algorithm adapted to a particular behavior to succeed a simple or complex goal. We have a method to emulate control instructions, operator and test equality to respect the algorithm according to goals.

## 8 Experiments and monitoring

Experiments are based on the behavior of the system. We observe the emergent knowledge (simple knowledge or variables with specific type and value) according to the goals. This observation allows us to know if the system is directed towards the programmed goals. To observe the system behavior, we have developed a graphical user interface describing the different links between variables of the system. We use a test platform with Aibo ERS7 by Sony to process different specifics scenario.

We have two experiments:

1. The robot has two goals and a set of experiences incompatible with the specified goals.
2. The robot has two goals and a set of experiences compatible with goals.

The aims of these experiments are to prove that a goal can be processed only if it is present in the ontology and in the set of experiences which define the assets of the robot ie, acquaintances between knowledge (a role for an agent). An experience has a name and a set of roles with different acquaintances.

We use Aibo recognition for objects, colors and persons. For the two experiments, the robot recognize these entities in the test environment: Alain, Mickael, Ball, Bone, Battery and Pink (two persons, two objects and one color, with a recognition rate of 90 per cent), and respect the four phases for the interpretation of data:

1. Data transit in the multi-agent system. At this phase, it is impossible for a human to interpret information, the behavior of the system.
2. System interpretation of information in order to create an emerging algorithm with morphology according to goals.
3. Choose a specific form with morphology. After this choice, specific roles will emerge.
4. Create an action plan with the emerging algorithm.

During the scenario processing, we name a focal point a set of emergent knowledge evolving in a time line according to sensors values.

### 8.1 First scenario

Features of the scenario:

- A tiny ontology (eleven mega-octet).
- More than one thousand agents evaluating more than seven thousands threads in the system.
- An unlimited activation coefficient<sup>1</sup>.
- A simple goal: to have pleasure (linked with Ball, Play, Alain, Cushion, Pink, Cover and Bone) or to have dissatisfaction (linked with Work, Generator, Alain and Fatigue).
- A set of experiences:
  - Play: Alain with acquaintances with ears, tailr<sup>2</sup>, ball, sleep, tenderness. Ball with acquaintances with frontof, jean-charles, ears, tailr1, led2, mouth2, sleep. Sleep with acquaintances with cushion, cover, work and play. Play with acquaintances with ears, tailr1, ball, keyboard and sleep.

<sup>1</sup>Agent number which can activate in a same time

<sup>2</sup>Tail can be moved on right(r) or on left(l)

- Play with peoples: Sympathy with acointances with love, alain, jean-charles, lionel, mickael. Love with acointances with alain and jean-charles. Frontof with acointances with ball and play. Bone with acointances with sit, tailrl, led9 and kiss.

Figure 5 shows emergent roles in the focal point for this experiment. We note there is no roles concerning the goal dissatisfaction because only one role is presents in the robot experience: the role Alain. The following focal point in the figure 6 shows others roles which increase the importance of the goal pleasure. In this experiment we note that the goal dissatisfaction can't be processed.

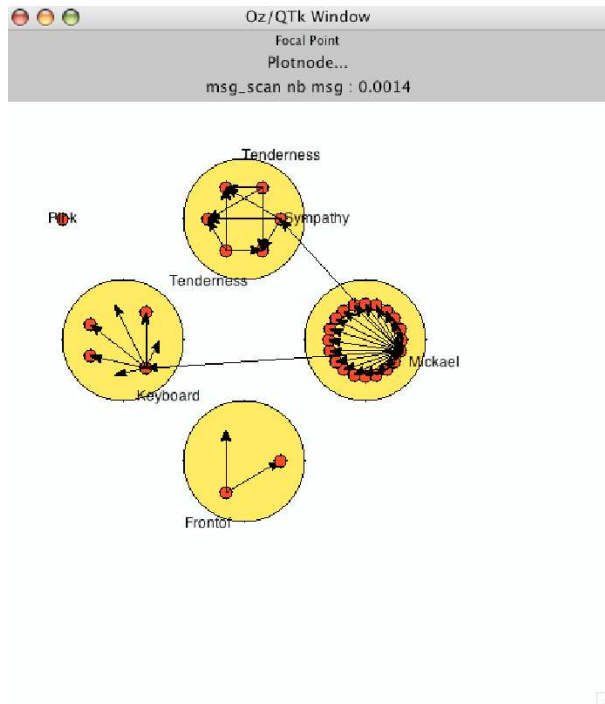


Figure 5: We note that goal dissatisfaction is not present on this focal point. The pleasure emerges according to inputs.

## 8.2 Second scenario

Features of the scenario:

- A tiny ontology (eleven mega-octet).
- More than one thousand agents evaluating more than seven thousands threads in the system.
- An unlimited activation coefficient.
- A simple goal: to have pleasure (linked with Ball, Play, Alain, Cushion, Pink, Cover and Bone) or to have dissatisfaction (linked with Work, Generator, Alain and Fatigue).
- A set of experiences with the addition of two experiences for this experiment (PlayBall and Work on computer):

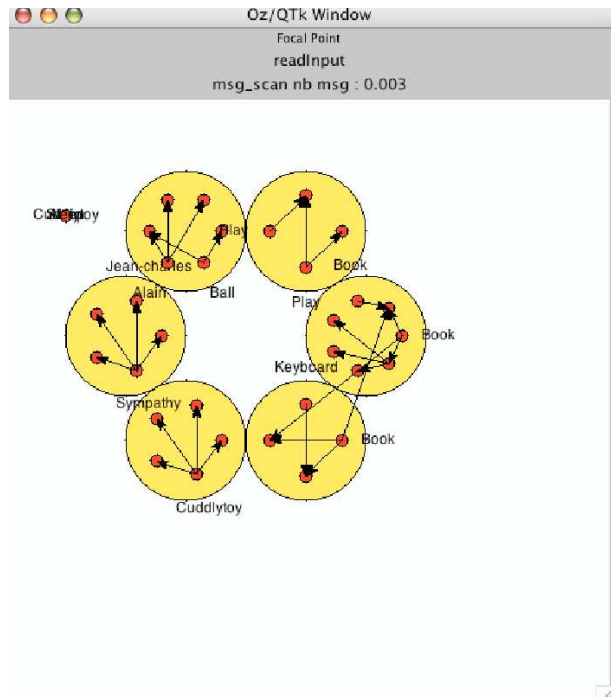


Figure 6: The pleasure emerges more and more. Without experience on Work, Generator, Alain or Fatigue, the goal dissatisfaction can't be processed. Here we have only the role Alain.

- Play: Alain with acointances with ears, tailrl, ball and sleep. Ball with acointances with frontof, jean-charles, ears, tailrl, led2, mouth2 and sleep. Sleep with acointances with cushion, cover and play. Play with acointances with ears, tailrl, ball, sleep and keyboard.
- PlayBall: Pink with acointances with ball, bone, keyboard and alain. Ball with acointances with frontof, jean-charles, ears, tailrl, led2, mouth2 and sleep.
- Play with peoples: Sympathy with acointances with love, alain, jean-charles, lionel and mickael. Love with acointances with alain and jean-charles. Frontof with acointances with ball and play. Bone with acointances with sit, tailrl, led9 and kiss.
- Work on computer: Keyboard with acointances with work, bear, play, mickael and kick. Book with acointances with bear, work and alain.

Figure 7 shows the first focal point for this experiment. We note that the role keyboard has an important emergence. It is logic because this role is present in goals, pleasure and dissatisfaction. Moreover, there is a strong link between this role and inputs of the system (mickael). For the role Bone, this role is present in the inputs and in the goal pleasure, so its emergence is immediate.

Figure 8 shows a new focal point. We note that the role Work which is only linked with the goal dissatisfaction

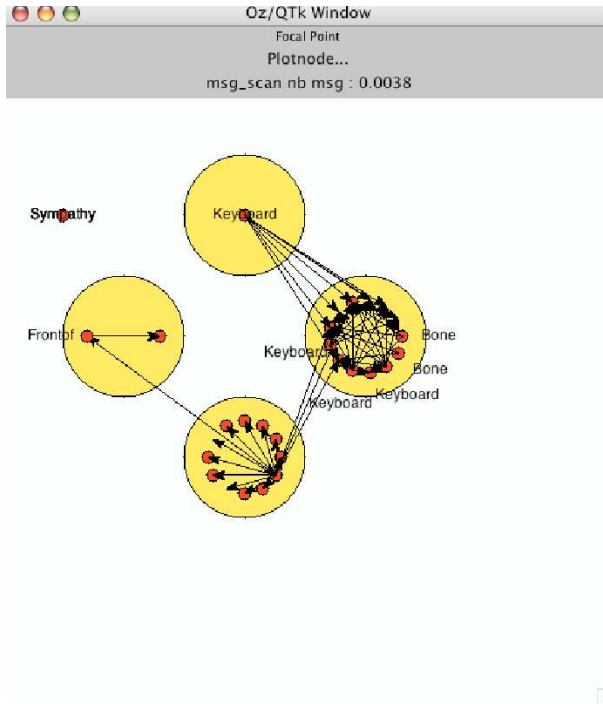


Figure 7: Keyboard is an important role in this focal point. The role Bone is also presents, it is in the goal pleasure.

tion emerges since it is present in the robot experience. This experiment shows that there is a strong link between experience and ontology. The robot can't use a knowledge to process a goal if this knowledge is not presents in an experience.

## 9 Decision-making and robot physical capacities

For each focal point presented in the previous section, there is a physical robot behavior. Several knowledge in the ontology are linked with physical capacities. So, if a decision is made to succeed a goal, values present in the form which are linked to physical capacities are sent to the robot. It is possible to the system to modify dynamically a physical movement to delete or update the precedent order.

The robot reacts dynamically when it recognizes an important element for a goal. It is difficult to see on the photo 9, but the robot moves its tail, ears, it smiles and it plays music. It is the behavior which is linked in the ontology and the experience of the system when the robot feels pleasure.

Currently, we have to work on a synchronisation between the decision-making and the processing of the physical behavior to have a physical action more precise to solve more complex goals.

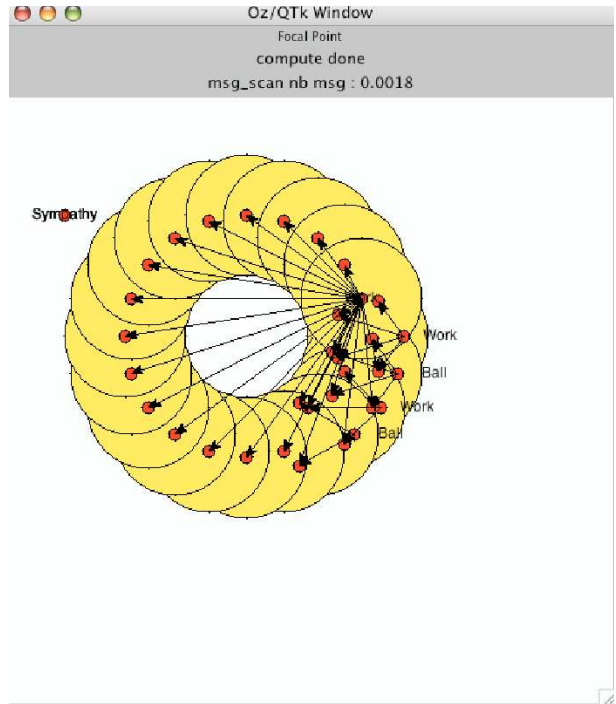


Figure 8: With a large experience, the focal point can be complex. We can see two opposed roles: Work and Ball. With this emergence, we note that the two goals, pleasure and dissatisfaction, can be processed with inputs of the system.



Figure 9: The robot has a specific behavior according to the focal point. If the emergent variables in the focal point are linked with physical capacities, the robot acts immediately. Here, the robot is happy, it moves its tail, ears and makes a sign with its eyes.

## 10 Conclusion

Industry needs flexibility in processing system. Robot market is more and more important and currently there is no system to manage a multirobots community. In this paper we present a programming method to build an abstraction layer to manage sensors, effectors, ontology and

goals system. This abstraction layer allow goals or ontology dynamic modification or, more difficult, the robot physical capacities modification. In experiments, we see a specific emergence of agents according to sensors values and the focal point (the current thought) of the system on a time scale. We see a specific direction according to the goals and different physicals actions on the robot. Now we can work on actions synchronisation to direct the robot precisely, make more experiments on multirobots, make experiment with a heigher ontology and distribute lot of agents on distant machines.

## References

- [1] R. A. Brooks. *Flesh and Machines*. Pantheon Books, 2002.
- [2] Rodney Brooks and Lynn A. Stein. Building brains for bodies. Technical Report AIM-1439, 1993.
- [3] J.C Campagne. *Morphologie et système multi-agent*. PhD thesis, Université Pierre et Marie Curie, 2005.
- [4] M. Camus and A. Cardon. Towards an emotional decision-making. In *Second GSFC/IEEE WRAC 2005 : Workshop on Radical Agent Concept, NASA Goddard Space Flight Center*, 2005.
- [5] A. Cardon, J.C Campagne, and M. Camus. A self-adapting system generating intentional behavior and emotions. In *Second GSFC/IEEE WRAC 2005 : Workshop on Radical Agent Concept, NASA Goddard Space Flight Center*, 2005.
- [6] Jean-Clair Poncet Christophe Guettier. Multi-levels planning for spacecraft autonomy. In *6th International Symposium on Artificial Intelligence and Robotic Applications for Space, Montreal*, 2001.
- [7] A. R. Damasio. *L'erreur de Descartes*. Odile Jacob, 1995.
- [8] R. Descartes. *Méditations métaphysiques*. Flammarion, 1997.
- [9] P.V. Roy and S. Haridi. *Concepts, Techniques, and Models of Computer Programming*. MIT Press, 2004.
- [10] R. Thom. *Modèles mathématiques de la morphogénèse*. Christian Bourgois, 1989.
- [11] R. Thom. *Paraboles et catastrophes*. Flammarion, 1999.