Modeling Autonomous Mobile System with an Agent Oriented Approach

Adil Sayouti, Hicham Medromi, Fatima Qrichi Aniba, Siham Benhadou and Abderrahim Echchahad

Team Architecture of Systems, ENSEM, BP 8118, Oasis, Casablanca, Morocco

Summary
This paper focuses on the design of a control architecture, via Internet, of a mobile system. The remote control extends the sensorimotor capacities and the possibilities of actions of a human being in a distant place. The Internet network (network without Quality of Service) limits the quantity of information that can be transmitted (bandwidth) and introduces delays which can make the remote control difficult or impossible. The solution proposed, through this work, to face the limitations of the Internet, is founded on the autonomy and the intelligence, based on multi-agents systems, granted to the mobile system in order to interact with its environment and to collaborate with the remote user. The need that consists in wanting to assign to the mobile system the autonomy and intelligence brought us to examine in the detail the choice of a control architecture. This paper will present the modeling process of an autonomous mobile system with an agent oriented approach.

Key words:
Control Architecture, Internet, Multi-agents System, Agent-Oriented Programming, AUML.

1. Introduction

Agent-based systems technology has generated lots of excitement in recent years because of its promise as a new paradigm for conceptualizing, designing, and implementing software systems. This promise is particularly attractive for creating software that operates in environments that are distributed and open, such as the internet.
An agent is a software entity which functions continuously and autonomously in a particular environment, often inhabited by other agents and processes. Agent is described as the software component exhibiting the following characteristics:

1. Autonomy: An agent has its own internal thread of execution, typically oriented to the achievement of a specific task, and it decides for itself what actions it should perform at what time.
2. Situatedness: Agents perform their actions while situated in a particular environment. The environment may be a computational one (e.g., a Web site) or a physical one (e.g., a manufacturing pipeline), and an agent can sense and effect some portions it.
3. Proactivity: In order to accomplish its design objectives in a dynamic and unpredictable environment the agent may need to act to ensure that its set goals are achieved and that new goals are opportunistically pursued whenever appropriate.

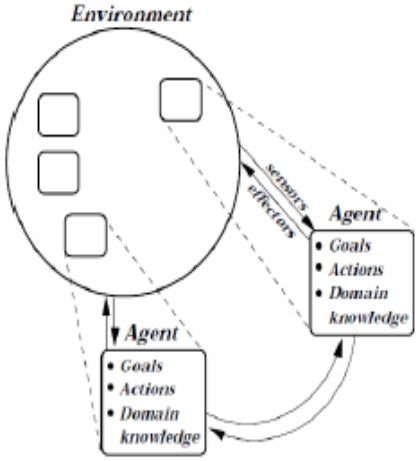

Fig. 1 Classical multi-agents behavior scheme

Agents can be useful as stand-alone entities that are delegated for particular tasks. However, in the majority of cases, agents exist in environments that contain other agents. In these multi-agents systems, the global behavior derives from the interaction among the constituent agents. Agents interact (cooperate, coordinate or negotiate) with one another, either to achieve a common objective or because this is necessary for them to achieve their own objectives [1].
Multi-agents systems (MAS) are based on the idea that a cooperative working environment comprising synergistic software components can cope with problems which are hard to solve using the traditional centralized approach to computation. Smaller software entities – software agents – with special capabilities (autonomous, reactive, pro-active and social) are used instead to interact in a flexible and dynamic way to solve problems more efficiently. Agents model each other's goals and actions; they may also

Manuscript received September 5, 2009
Manuscript revised September 20, 2009

interact directly (communicate). The classical multi-agents behavior scheme is presented in Figure 1.

This paper is presented as follows: on the next section, we describe our control architecture based on multi-agents system to face the limitation of the Internet. In section 3, we present the modeling process of an autonomous mobile system task with Agent UML. In section 4, we present an application of control of an autonomous mobile system (The Khepera III robot) as an illustrative example. Some conclusions are presented in section 5.

2. Control Architecture Proposed

Robot control architectures are sophisticated control systems with the purpose of enabling robots to do usual physical work in the real world. Figure 2 depicts a very simplified view of robots control architectures. The robot control system continuously perceives the environment through sensors, updates an internal model of the world, deliberates, and passes actuator parameters to the hardware interface in order to pursue given goals.

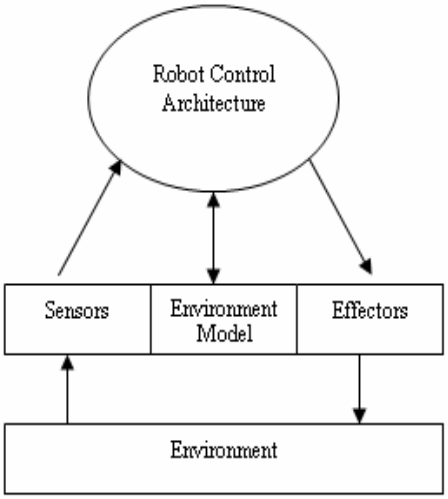

Fig. 2 Simplified Robot Control Architecture

The designer has to choose the way he will give autonomy to his robot. He has mainly two orientations: “reactive” capacities or “deliberative” capacities [2]. These two capacities are complementary to let a robot perform a task autonomously (Figure 3). The designer must built a coherent assembly of various functions achieving these capacities.

One of the first author who expressed the need for a control architecture was R.A. Brooks [3]. In 1986, he presented an architecture for autonomous robots called “subsumption architecture”. Then other various architectures were developed based on different approaches, generally conditioned by the specific robot application that the architecture had to control, we can mention: The architecture 4-D/RCS developed by the Army Research Laboratory [4], CLARAty Architecture [5], LAAS Architecture (Laas Architecture for Autonomous System) [6], AuRA Architecture (Autonomous Robot Architecture) [7], DGA Architecture [8], The DAMN architecture (Distributed Architecture for Mobile Navigation) [9], etc.

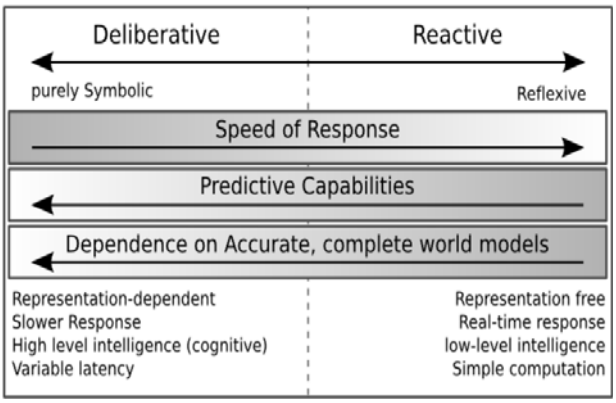

Fig. 3 Control architecture design approach

From the study of the different control architectures, we propose a hybrid control architecture, called EAAS for EAS Architecture for Autonomous system, including a deliberative part (Actions Selection Agent) and a reactive part.

Hybrid deliberative/reactive architectures emerged as a result of the recognition that there is an appropriate use of symbolic knowledge in the formulation of robot behaviors. Such systems take advantage of a priori knowledge of the environment to formulate correct behavioral sets that can be used during run-time. Representations are used during plan formulation but not during plan execution. This creates greater exibility for a reactive robot by allowing a high-level deliberative planner to configure the robot's behaviors in accordance with the task at hand, known or anticipated environmental conditions, and available robotic resources.

Our control architecture [10] is made up of two parts (Figure 4). The deliberative part which contains a path planner, a navigator and a pilot. The reactive part is based on direct link between the sensors (Perception Agent) and the effectors (Action Agent). The hardware link agent is an interface between the software architecture and real robot. Changing the real robot require the use of a specific agent but no change in the overall architecture.

Fundamental capacities of our architecture encompass autonomy, intelligence, modularity, encapsulation, scability and parallel execution. To fulfil these requirements, we decided to use a multi-agents formalism

that fits naturally our needs. The communication between agents is realized by messages. Object oriented language is therefore absolutely suited for programming agents (we chose java). We use threads to obtain parallelism (each agent is represented by a thread in the overall process).

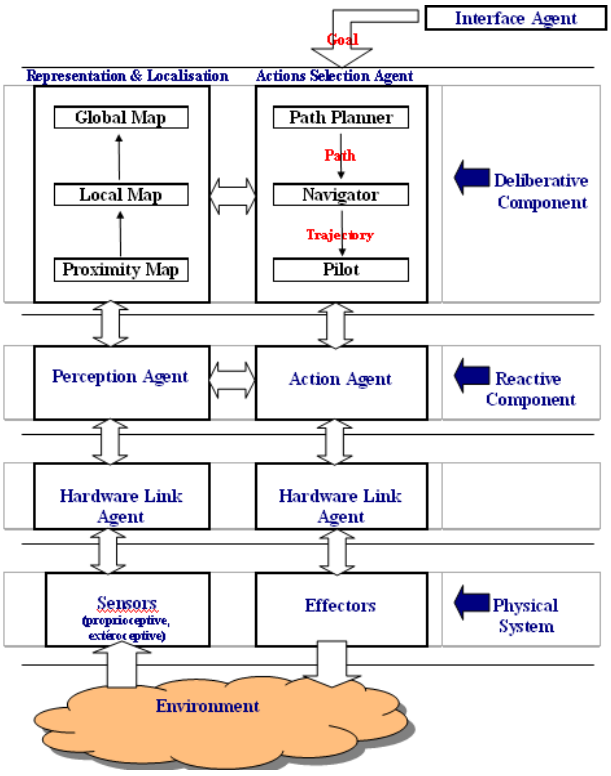

Fig. 4 EAAS Architecture proposed

3. Modeling the Navigation Task of a Autonomous Mobile System with AUML

3.1 Agent UML

UML [11] is certainly the best known and most widely used modeling languages. Agent UML is the most famous graphic modeling language to describe multi-agent systems.

The modeling language Agent UML [12] [13] inherits from UML and integrates the differences between agents and objects [14]. Agent UML is an extension of UML to take into account the concepts officer that he did not. Because Agent UML is an extension, it inherits the representations proposed by UML. Here is the list: Sequence diagrams, Collaboration diagrams, Activity diagrams, Statechart, Use case diagrams, Class diagrams, Object diagrams, Packages, Component diagrams and Deployment diagrams. The first five correspond to representations of dynamic charts, while the last four correspond to static diagrams.

The sequence diagrams have been modified since they bear the name of protocol diagrams and correspond to the representation of interaction protocols. The protocol diagrams describe the interaction protocols used by agents. They exploit the wealth of communications between agents. The collaboration diagrams include both the class diagrams and protocol diagrams. These diagrams allow to group the structure of the system with class diagrams of the system behavior through the exchange of messages with protocol diagrams. The activity diagrams and interaction diagrams have similarities because they both deal of system activity. The activity diagrams describe the interaction between the activities while the interaction diagrams are among the objects. The statecharts similar to activity diagrams except that it is not to model activities, but states of the system. The statecharts are used to represent the behavior of the system. The use case diagrams are used to define use cases of the system and an analysis of the system. The class diagrams have also been modified. We must not forget that an agent differs from an object, therefore, the class diagram that can represent the classes of the system will be modified. The class diagrams describe the structure and relationships between the various classes of the system. The object diagrams are instantiated versions of class diagrams. These are generally the objects represent a specific moment of the execution of the system to know the values of attributes. The packages are a mechanism for organizing elements into groups. The component diagrams describe the components needed for implementing the system. A component is a physical element of the system that provides services through interfaces. The deployment diagrams are physically organizing the physical system.

In the next section of this paper, one using our control architecture, we will present the modeling process of an autonomous mobile system task with AUML. The modeling process starts with a Business Process Model. Then we will present the agents class diagram that describe the solution proposed.

3.2 Business Process Model of a Mobile System Navigation Task

We choose the navigation task to be agent-oriented modeled. In this task, the mobile system has a goal, which is a place where it has to go. The problem consists in self-localization through all the way in relation to the goal. The information that is provided to the system is the direction and the distance of the goal to the mobile system. Thus the mobile system must cope with errors in the movements through the readings of its sensors.

Figure 5 shows the business process model diagram representing how the mobile system navigation task must work. The actions selection agent of our control architecture was developed according to the hierarchical paradigm, i.e. the mobile system acts only after planning through processing the sensor readings. This control architecture enables the mobile system to do a navigation task with continuous sensing in a static environment where there is no obstacle moving. While the task is being accomplished the mobile system does a cyclic process that start with new information of its sensors and finishes with a movement produced by the effectors.

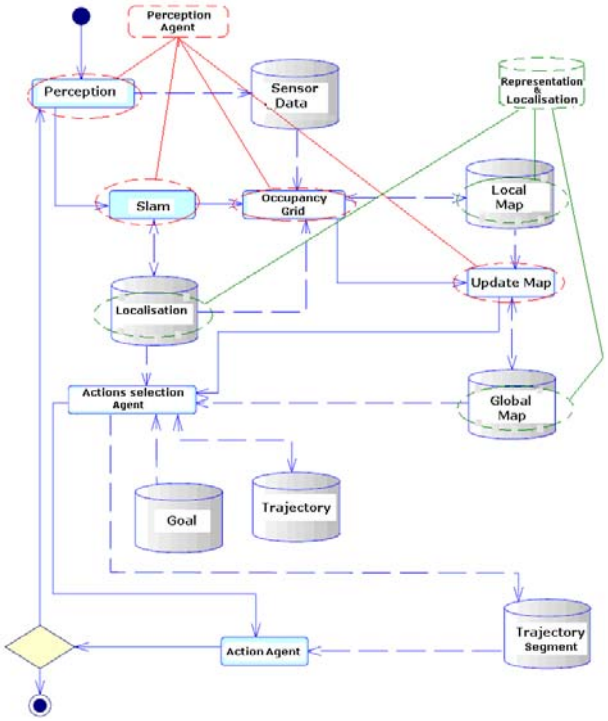

Fig. 5 Business process model diagram of a mobile system navigation task

When the navigation task starts, the first process (processes are indicated in the diagram as boxes) executed is the Perception that updates the resource Sensor Data (resources are represented by cylinders) according the information retrieved by range sensors in the mobile system. Sensor Data is a list of spatial coordinates. These coordinates are the ending points of the vectors starting in the mobile system and ending in the detected obstacles.

After Perception another process named SLAM is executed. SLAM (Simultaneous Localization And Mapping) is a statistical framework for simultaneously solving the mapping problem and the induced problem of localizing the mobile system relative to a growing map. As the mobile system is navigating toward its goal, it needs to use information of its sensor for self-localization. This allows the mobile system to correct its trajectory and to use an estimative of its localization to incorporate a local map in the global one. Basically the SLAM algorithm is an estimator that can integrate information of several sensor sources to obtain a better estimative of the mobile system localization in space. In the diagram, the SLAM process accesses the resource Localization and updates it based on the new Sensor Data.

With Localization and Sensor Data updated, the Occupancy Grid process is called. The Occupancy Grid transforms that list of coordinates (Sensor Data) in a tessellated map of probability of obstacles occupying a specific cell. It accesses the resource Local Map and updated it. This map can be represented as a matrix where the value of each element, or cell, represents the probability of occupation in an area of the space. After the Occupancy Grid provides an updated Local Map, the process Update Map incorporates the Local Map in the Global Map.

Then the process Actions Selection Agent is called and provided with the updated Global Map, with the resource Goal of the navigation task and with the actual localization of the mobile system. The Actions Selection Agent produces a resource named Trajectory that is the whole trajectory to be followed by the mobile system in order to achieve the goal position. The trajectory can be represented as a sequence of coordinates interpolated by straight lines. Each line is a trajectory segment of the whole trajectory.

Besides the whole trajectory, the Actions Selection Agent also provides and updates the resource Trajectory Segment. This resource gives the current line segment to be followed by the mobile system. Finally, the process Action Agent uses the Trajectory Segment resource to send commands to act the mobile system. While the mobile system does not reach the resource Goal provided externally by a user, a new cycle begins.

The business process model allows us to describe a general solution for the problem. In this step, the main processes and information used in the solution are specified. In the next section, we will present the agents class diagram which is a static representation of the system showing the class and their relationship.

3.3 Agents class diagram

Considering the business process model of Fig. 5, an agents class diagram was designed and it is showed in Fig.6. Below it is a description of the designed agents classes [15].

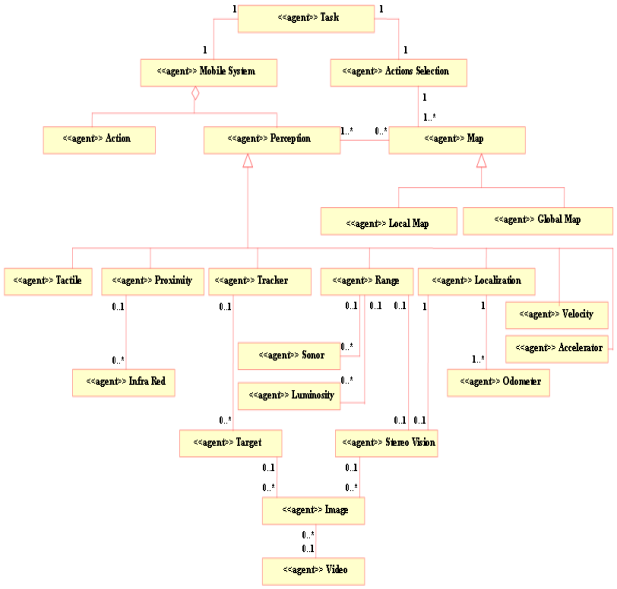

Fig. 6 Agents class diagram developed

Task is the class responsible for specifying the task to be performed by the mobile system. As we are interested in a navigation task using a single mobile system and a hierarchic approach for the deliberative part, the Task class will make use of one agent of the Mobile System class and one agent of the Actions Selection class. The main attribute of the Task class is the goal. Mobile System is the class responsible for describing the mobile system that will be used in the task. The Mobile System class is made of a composition of at least one Action and at least one Perception. Action is the base class responsible for interfacing with the effectors of the mobile system. Perception is a base class from which many perception classes can be derived. Some of the possible perception classes are represented in the agents class diagram and they are Tactile, Proximity, Tracker, Range, Localization, Velocity and Accelerator. Localization, Velocity and Accelerator classes are responsible for giving internal information about the actual state of the mobile system. The Localization class can be based on the readings of odometers to estimate the mobile system localization. Range class represents a range sensor in the mobile system. It provides an estimative of the direction and distance between the mobile system and the obstacles. Proximity is the class that represents the proximity sensor of the mobile system. A proximity sensor can detect when an object is close to the mobile system. Tracker is the class responsible to describe a visual tracking sensor. Through this sensor is possible to track an object in an image and to retrieve some information about the visual changes suffered by this object. Stereo Vision is one of the classes that can be used by the Range sensor class. Stereo Vision is responsible to acquire and process one pair of images and to provide distance estimation from obstacles. The two images required must be taken from two different viewpoints. Image is a class to support methods for image processing and image manipulation. Actions Selection is the class responsible for the trajectory planning algorithms used in the navigation task. The Actions Selection class uses the Map class to take decisions about the trajectory that the mobile system should follow. The main attribute of the Actions Selection class is the Goal position set by the Task. Map is the class responsible for storing and manipulating the maps. There are two classes derived from it that represent the two different types of map: Local Map and Global Map. Both local and global maps can represent the map using occupancy grids [16], and the current robot localization is indicated on the maps. The Local Map has a method called Slam that uses SLAM algorithm that allows correcting the mobile system localization. The Global Map class has a method called Update that allows the local map to be fused on the global map.

4. Application

During the project development, different configurations were tested in different environments. The aim is to develop a more reliable system framework that can be used in the real world.

Fig. 7 Khepera in its environment

The main advantages of the robot Khepera III are the small size and the very long power autonomy of 8 hours. As shown in Figure 7, our Khepera mobile robot was controlled to explore its environment while avoiding several static obstacles in order to reach its goal.

5. Conclusion

The first part of this paper presented our control architecture, based on multi-agents system, to assign to the mobile system the autonomy and the intelligence in order to interact with its environment. The second part showed

an application on mobile system navigation modeled using agent-oriented concepts. In this navigational task the localization of the mobile system is corrected through SLAM and the planning of actions is done using maps based on occupancy grids. There are still many open problems to solve, until exploration of unknown or very complex environments through the Web becomes reality.

References

[1] Ferber, Jacques, Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence, Addison Wesley Longman, Harlow, UK, 1999.

[2] Cyril Novales, Gilles Mourioux, Gerard Poisson, A multi-level architecture controlling robots from autonomy to teleoperation. First National Workshop on Control Architectures of Robots. Montpellier. April 6,7 2006.

[3] R.A. Brooks, A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, vol. 2, n°. 1, pp. 14–23, 1986.

[4] J. S. Albus, 4D/RCS A Reference Model Architecture for Intelligent Unmanned Ground Vehicules, Proceedings of the SPIE 16th Annual International Symposium on Aerospace/Defense Sensing, Simulation and Controls, Orlando, FL, April 1 – 5, 2002.

[5] R. Volpe and al. CLARAty: Coupled Layer Architecture for Robotic Autonomy." *JPL Technical Report D-19975*, Dec 2000.

[6] R. Alami, R. Chatila, S.Fleury , M.Ghallab and F.Ingrand: An architecture for autonomy. *The International Journal of Robotics Research*, Special Issue on Integrated Architectures for Robot Control and Programming, vol. 17, no 4, pp. 315-337, 1998.

[7] R.C. Arkin, Behavior-based Robotics. MIT Press, 1998

[8] A. Dalgalarrondo, Intégration de la fonction perception dans une architecture de contrôle de robot mobile autonome. Thèse de doctorat, Université Paris-Sud, Orsay, 2001.

[9] J. Rosenblatt, DAMN: A Distributed Architecture for Mobile Navigation. In proceedings of the 1995 AAAI Spring Symposium on Lessons Learned from Implemented Software Architectures for Physical Agents, AAAI Press, Menlo Park, CA, 1995.

[10] A. Sayouti, F. Qrichi Aniba, H. Medromi. "Remote Control Architecture over Internet Based on Multi agents Systems". International Review on Computers and Software (I.RE.CO.S), Vol 3, N. 6, pp. 666 – 671, November 2008.

[11] G. BOOCH, J. RUMBAUGH, I. JACOBSON, The Unified Modeling Language User Guide, Addison-Wesley, Reading, Massachusetts, USA, 1999.

[12] B. BAUER, J.P MULLER, J. ODELL, «An extension of UML by protocols for multiagent interaction», International Conference on MultiAgent Systems (ICMAS'00), Boston, Massachussetts, july, 10-12 2000.

[13] J. Odell, V. D. Parunak and B. Bauer, « Extending UML for Agents », Proc of the AOIS at the 17th National conference on Artificial Intelligence, Austin, Texas, july, 30 2000.

[14] N. R. JENNINGS, M. WOOLDRIDGE, «Agent-Oriented Software Engineering», BRADSHAW J., Ed., Handbook in Agent Technology, MIT Press, 2000.

[15] A. SAYOUTI, "Conception et Réalisation d'une Architecture de Contrôle à Distance Via Internet à Base des Systèmes Multi-Agents", Phd. Thesis, ENSEM, Hassan II University 2009.

[16] A. Elfes, "Occupancy Grids: A probabilistic framework for robot perception and navigation", Ph.D. Thesis, Carnegie-Mellon University, 1989.

Adil Sayouti received the PhD in computer science from the ENSEM, Hassan II University in July 2009, Casablanca, Morocco. In 2003 he obtained the Microsoft Certified Systems Engineer **(**MCSE**)**. In 2005 he joined the system architecture team of the ENSEM, Casablanca, Morocco. His actual main research interests concern Remote Control over Internet Based on Multi agents Systems.

Hicham Medromi received the PhD in engineering science from the Sophia Antipolis University in 1996, Nice, France. He is responsible of the system architecture team of the ENSEM Hassan II University, Casablanca, Morocco. His actual main research interest concern Control Architecture of Mobile Systems Based on Multi Agents Systems. Since 2003 he is a full professor for automatic productic and computer sciences at the ENSEM, Hassan II University, Casablanca.

Fatima Qrichi Aniba received an electrical Engineer's degree from the ENSAM in 2003 Meknes, Morocco. In 2005 she got her Degree in High Education Deepened in automatic productic from the ENSEM, Hassan II University, Casablanca, Morocco. In 2005 she rejoined the system architecture team of the ENSEM. Her main research is mainly about Real Time Architecture Based on Multi Agents Systems.

Siham Benhadou received her Degree in High Education Deepened in computer science in 2004 from the HASSAN II University, Casablanca, Morocco. In 2006 she joined the system architecture team of the ENSEM, Casablanca, Morocco. Her actual main research interests concern Intrusion Detection System Based on Multi agents Systems.

Abderrahim Echchahad received his master in system architecture from the ENSEM in 2009, HASSAN II University, Casablanca, Morocco. In the same year he joined the system architecture team of the ENSEM, Casablanca, Morocco. His actual main research interests concern Control of Robotic Systems Based on Multi agents Systems.