International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 5 – No. 5, April 2013 – www.ijais.org

Multi-Agents Systems – Modeling, Programming and Applications

Adil SAYOUTI, Hicham MEDROMI
Team Architecture of Systems
LISER - Laboratory
ENSEM, Hassan II University
BP 8118, Oasis, Casablanca, Morocco

Faissal ELMARIAMI, Abdelaziz BELFQIH
Team Electrical Systems
ENSEM, Hassan II University
BP 8118, Oasis, Casablanca, Morocco

ABSTRACT
A multi-agents system is a system composed of multiple interacting intelligent agents who can be used to solve problems which are difficult or impossible for an individual agent or monolithic system to solve. Multi-agents systems are open and extensible systems that allow for the deployment of autonomous and proactive software components. This is the reason why they are brought up and used in several application domains. In this paper, we present the application of the multi-agents systems in the remote control, network security and telecommunication domains. Those applications are realized by the System Architecture Team of the ENSEM, Hassan II University.

In the first section of this paper, we present the multi-agents approach. In the second section, we describe different architectures, based on multi-agents systems, proposed by the system architecture team of ENSEM, Hassan II University. In the third section, we present a realization in order to validate our architectures and the choice of the multi-agents approach.

Keywords
Multi-Agents System, Distributed System, Autonomous and Intelligent System, Mobile System, Telecommunication, Network Security.

1. INTRODUCTION
Agent-based computing represents a novel software engineering paradigm that has emerged from merging two technologies [1], namely artifcial intelligence (AI) and object oriented distributed computing [2]. Agent-based systems aim to strike a balance between artificial intelligence and computational utility.
Agents are intelligent, autonomous, software components capable of interacting with others within an application, attaining a common goal and thereby contributing to the resolution of some given problem. They are important because they inter-operate within modern applications like remote control, telecommunications, network security [3] and electronic commerce.
Over the years, a wide range of software engineering paradigms have been devised (e.g., procedural programming, structured programming, declarative programming, object-oriented programming, design patterns, application frameworks and component-ware) to deal with the increasing complexity of software applications. Although each successive development claims to make the engineering process easier, researchers continually strive for more efficient and powerful software engineering techniques, especially as solutions for ever more demanding applications are required. Most real-world applications of today are significantly more complex than before as they contain many dynamically interacting components, each with its own thread of control. Most software engineering paradigms are unable to provide structures and techniques that make it easier to handle this complexity. Consequently a lot of research has now been directed toward treating computation as a process of interactions. Tools and technologies have been developed to understand, model, and implement systems in which interactions are the norm.
Furthermore, software development has now become a knowledge-intensive activity. Current software representations (from modeling to programming languages) are non-intentional. They are meant to record the results of software work but not the process or the reasoning behind them. Thus there is a reason to develop a framework of software engineering that accounts for the intentional dimensions, namely intents and motivations, goals and reasons, alternatives, beliefs and assumptions in its methodologies.
Against this background, we will argue that analyzing, designing, and implementing software as a collection of interacting intelligent agents represents a promising approach [4] to software engineering. An agent is an encapsulation of goals, know-how and resources. Agent-oriented techniques provide a natural way for modeling complex systems, by decomposing its problem space into autonomous agents and their interactions. Moreover, they enhance the reliability and reduce the cost and time-to-market of software applications by allowing their development through the assembly of a set of reusable software agents.

In the next section, we make a strong case for agent-oriented approach for software engineering and advance our arguments by comparing their effectiveness against object-oriented approach.

2. MULTI-AGENTS APPROACH
2.1 Agent characteristics
A characteristic is an intrinsic or physical property of an agent. The following are some common agent characteristics [5]:

- Autonomy: An agent can act on another's behalf without much guidance.
- Communication: An agent can communicate with other agents on a common topic of discourse by exchanging a sequence of messages in a speech-act-based language that others understand. The domain of discourse is described by its ontology.

- Mobility: An agent can migrate from one system to another in a pre-determined fashion or at its own discretion. Accordingly, agents can be static or mobile.
- Learning: An agent can have the ability to learn new information about the environment in which it is deployed and dynamically improve upon its own behavior.
- Cooperation: An agent can collaborate and cooperate with other agents or its user during its execution to minimize redundancy and to solve a common problem.

In the next section, we present the main differences between an agent and an object

2.2 Approach agent vs. approach object

The multi-agents system is considered as an object-oriented system that is associated to an intelligent meta-system. By this way, an agent is viewed as an object that has a layer of intelligence, comprising a number of capabilities such as uniform communication protocol, perception, reaction and deliberation, all of them not inherent to objects. However, the Agent oriented approach (AOP) has code, states and agent invocations. The agents also have individual rules and goals to make them appear like active objects within initiative. In AOP the class is replaced by role, state variable with belief/knowledge and method with message. The role definitions describe the agent capability and the information needed to desired results. In order to the agents act with intelligence in their environment, the idea is to develop the complex entities and provide the agents with the knowledge and beliefs to be able to achieve their desires.
The table 1 illustrates the differences between the agent approach and the object approach.

Criteria	Agent approach	Object approach
Basic Unit	Agent	Object
Unit state	Mental components	Unconstrained
Communication paradigm	Peer-peer	Client-server
Communication mode	Message passing	Message passing
Communication type	Local (mobile) + remote (static)	Mostly remote
API	Uniform method call	Unconstrained
Method constraints	Honesty, consistency, etc.	None
Message type	Speech Acts (ACL)	Unconstrained
Mobility	Autonomy and mobility-related metadata	No autonomy or mobility related meta data
Inheritance	Mental states	Methods and attributes
Intelligence	Intelligent operations	Not always present

Table 1. Agent approach vs. object agent

The cooperation in a multi-agents system is based on the communication and the interaction between agents. This axis will be detailed in the next section.

2.3 Agent communication

The agent communication, also known as the agent-based messaging paradigm [6], provides a universal messaging language with a consistent speech-act-based, uniform messaging interface for exchanging information, statically or dynamically, among software entities. Agent communication has the following advantages over the traditional client-server (RPC) based communication:

- De-centralized, peer-peer communication, as opposed to the traditional client-server roles
- Asynchronous exchange of messages
- Universal message-based language with speech-act-based interface
- Single method invocation for all types of message exchanges (FIPA : Foundation for Intelligent Physical Agents)

The communication involves at least two parties: a sending agent that generates the information and transmits it and a receiving agent that receives the message and uses the information.
The information that is exchanged between the communicating parties may be formally coded into a universally understood agent communication language (ACL) with a speech act based interface. The sending agent on generating this ACL coded message string invokes the message method of the recipient and passes the string through it (FIPA framework). The receiving agent, on receiving this message, decodes the information and then performs the necessary actions. In case of a bidirectional communication, it may communicate the result back to the sender by reciprocating the same process.

2.4 Agent Communication Language

Agent communication, under this paradigm, is accomplished through the use of three components: ontology, content language, and agent communication language. Ontology enumerates the terms comprised by the application domain. The content language is used to combine terms in the ontology into meaningful sentences in the language as defined by the grammar. Sometimes the two are so tightly coupled that they become one. Finally, the agent communication language acts as a medium for exchanging dialogs among agents, containing sentences of the content language. It provides the outer encoding layer, which determines the type of agent interaction, identifies the network protocol with which to deliver the message, and supplies a speech act also known as communicative act or performative. The communicative act indicates whether the message is an assertion, a query, a command or any other acceptable speech form. ACLs range from some form of primitive communication to elaborated standards. Two of the most widely used ACLs are knowledge query manipulation language (KQML) and FIPA ACL [7]. Knowledge interchange format (KIF) is often used as a content language with KQML. Likewise, semantic language (SL) is often used to represent the application domain, even though the FIPA ACL specification document does not make any commitment to a particular content language.

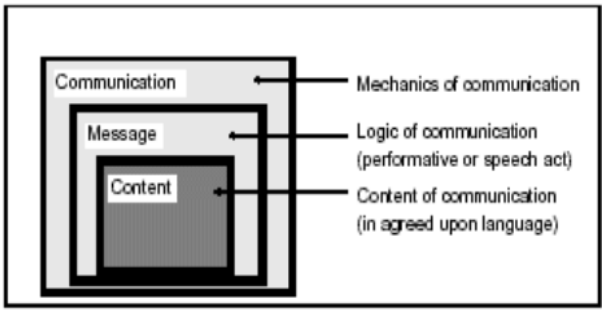

Fig. 1. The three layers of a KQML message

2.5 Interaction Protocols

Although FIPA uses AUML [8] to represent its standard interaction protocols, we use colored Petri nets (CPNs) [9], because their formal properties facilitate the modelling of

concurrent conversations in an integrated fashion. The availability of net analysis tools, means that it is possible to check the designed protocols and role interactions for undesired loops and deadlock conditions, and this can then help eliminate human errors introduced in the design process.

Figures 2 and 3 shows the representation of the FIPA request interaction protocol. Each interaction protocol is modeled in terms of the individual agent roles in the interaction: for each individual role there is a separate Petri net. The collection of individual Petri nets associated with all the relevant roles represents the entire interaction protocol. For every conversation, there are always at least two roles: that of the initiator of the conversation and the roles of the other participants in the conversation.

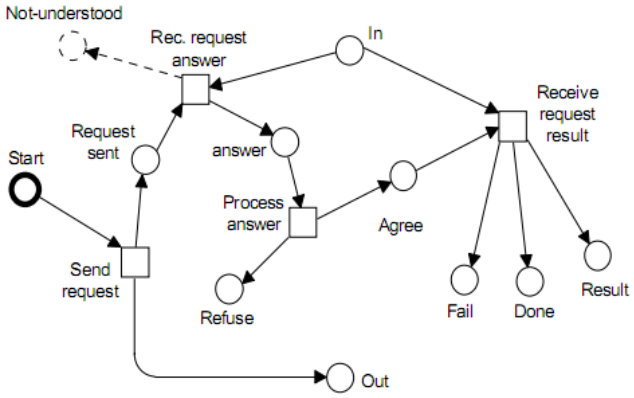

Fig. 2. Request interaction protocol for the Initiator role

Figure 2 depicts the initiator of the FIPA request interaction, and Figure 3 shows the Participant interaction. For diagrammatic simplicity, we omit the inscriptions from the diagram, but we will describe some of them below. The In place (in this and the following Petri net diagrams) will have tokens placed there when the agent receives messages from other agents.

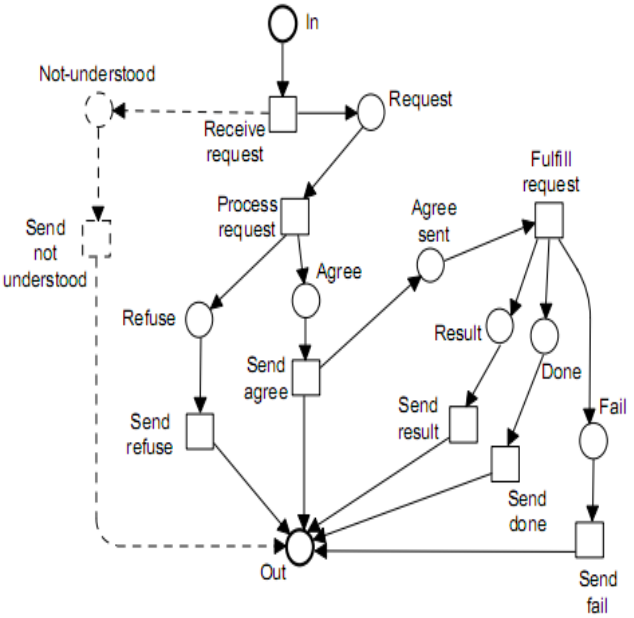

Fig. 3. Request interaction protocol: the Participant role

The In place is a fusion node (a place common to two or more nets): the very same In place may exist on other Petri nets that also represent conversations in which the agent may be engaged. When the agent receives a message from another agent, a token with information associated with the message is placed in the In place, which may be shared by several Petri nets. The transitions connected to the In place have guards on them such that the transitions are only enabled by a token on the In place with the appropriate qualification.

The Initiator of the request interaction will have a token placed in the Start place, and this will trigger the Send request transition to place a token in the Out place. We assume that the communication transport machinery causes tokens to disappear from a Petri net's Out place and (usually) a corresponding token to appear on the In place of another agent. The transfer may not be instantaneous, or even guaranteed to occur; it is possible for a token to disappear from one role's Out place without a corresponding token appearing at another agent's In place.

Note that the Initiator could be involved in several concurrent request interaction conversations, and the placement of specific tokens in the Agree place enables this agent to keep track of which responses correspond to which conversations. This shows how the colored Petri net representation facilitates the management of concurrent interactions involving the same protocol.

In the next section, we present the application of multi-agents systems in different fields. In the first time, we describe the different architectures. Then, we present some realizations.

3. THE PROPOSED ARCHITECTURES BASED ON MULTI-AGENTS SYSTEMS

In this section, we present the architectures proposed by the system architecture team. We are interested in this section to the following areas: Mobile systems [10], Telecom and network security.

3.1 Remote control on Internet domain

Our architecture EAAS1 (version 1), for remote control over Internet [11], consists in five agents: interface agent, actions selection agent, perception agent, action agent and hardware link agent.

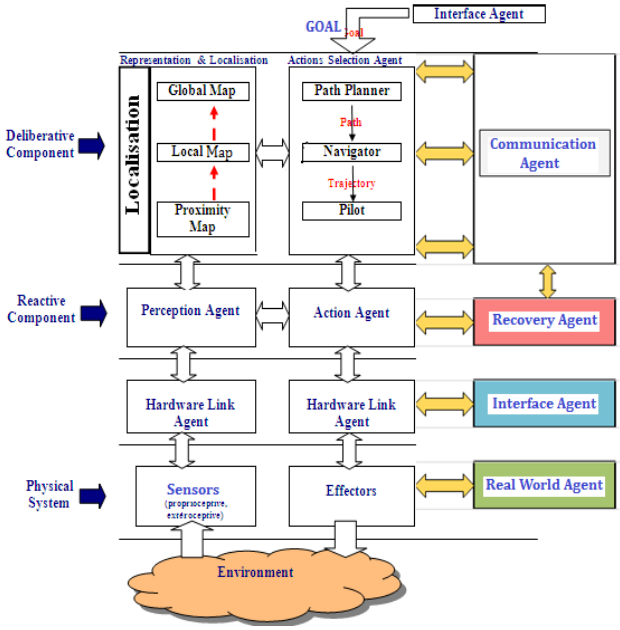

Fig. 4. The proposed Architecture - EAAS2

The interface agent is the high level of our control architecture. It must generate a succession of goal, or missions for the actions selection agent, according to the general mission of the mobile system. It is the "ultimate" mobile system autonomy concept: the mobile system generates itself its own attitudes and its own actions by using its own decisions. The perception agent manages the processing of incoming data (the sensor measurements) and creates representations of the environment. The actions selection agent must choose the robot behavior according to all information available and necessary to this choice: the fixed goal, representations and the robot localization.

In order to give to the remote user the possibility to tele-operate the mobile system. This architecture gives to the operator the possibility to communicate with the physical system. New agents have been added to the EEAS1. The EAAS2 (version 2) [12] consists in the EAAS1 and the agents: communication agent, recovery agent, interface agent and real world agent.

3.2 Telecommunication domain

The architecture proposed [13] in this work is a multi agent architecture, in which each agent is autonomous and able to cooperate, coordinate and communicate with other agents intelligently to achieve the system task. The agents can be reactive or cognitive. They are provided with two functions, the communication and the realization of its own task. The task of communication consists of passing information to the other agents or simply to relieve messages for the other agents. The specifics tasks consist of checking, trying, normalizing data or make decision… The SMS gateway which is the essential element of our platform appears in the form of two under multi agent system called sensor management and Robot control. These under system collaborate between them in a continuous way to command the robots and to assure an adapted communication to the need of our platform. Bellow a description of the gateway multi agent systems. Our architecture consists in six agents: interface agent, perception agent, management agent, Learning agent, action selection agent and action agent.

- The interface agent: this agent is the high level of our control architecture. It must generate a succession of goal as the objective coordinates, the physical data to be captured ... The Gateway generates itself the robot attitudes and actions as well as the radio management communication of the robots by using its own decisions
- The Management agent: this agent analyzes the data received by the user, if they are correct he launches the proprio/extro sensors to localize the robots then, the connexion sensors to detect the radio communication media existent in each robot and the battery level of these robots. Data captured are then transmitted to the perception agent that analyze and normalize it for creating a representation of environment. The management agent decide depending on the content of the representation, it's knowledge base and the location of the objective, which robot he will activate and which radio communication he will use to communicate with this robot. When the choice is made, the agent disables the radio communication that will not be used by the gateway and the robot. All decisions taken by the management agent is stored in its knowledge base, and an update of the representation is made.
- The learning agent: this agent made the connection between the management agent and the knowledge base (BC). BC contains all the rules necessary to the box for making his decision (Robot to activate, radio communication to choose….) we can also find the history of past orders ;this history allow the management agent to define directly and rapidly the action plan of the robot without redoing the representation of the environment.

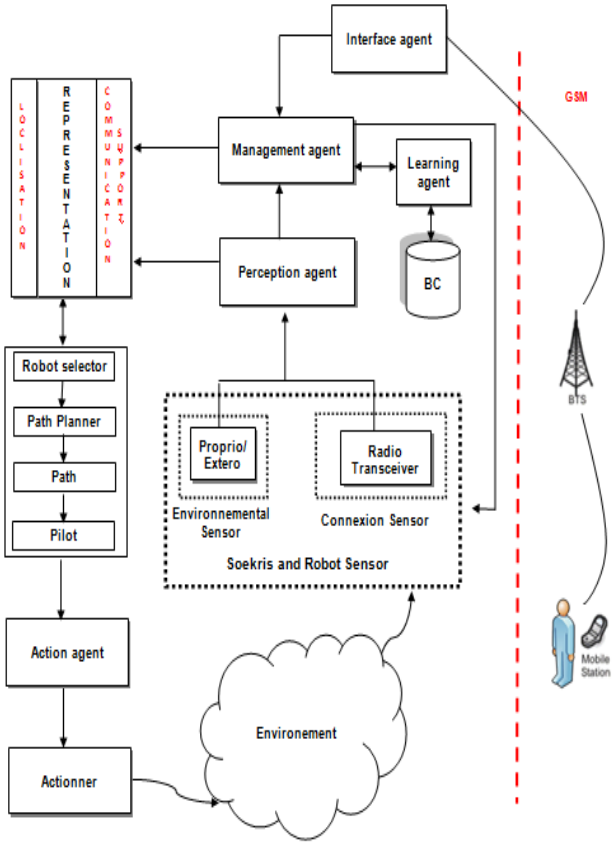

Fig. 5. Gateway multi agent architecture

- The actions selection agent must choose the robot behavior according to all information available and necessary to this choice: the fixed goal, representations and the robot localization. The actions selection agent contains a path planner, a navigator and a pilot. The path planner may take a goal as input and give a path for achieving the goal as output. The Navigator must translate a path into a trajectory for the pilot. The path does not take into account physical constraints of the robot, but the trajectory that it delivers must integrate them. The function of the pilot is to convert this trajectory into orders to be performed by the action agent.
- The action agent consists of a set of behaviors controlling the robot effectors.

3.3 Architecture of proposed platform of intrusion detection

The proposed architecture [14] consists of several agents distributed at different network points with different roles.

The proposed intrusion detection architecture consists of several agents, monitoring the network or sensitive positions, with the following characteristics:

The analyzer based on a distributed approach, using multi-agent system, includes: Agents; responsible for collecting sensor data exchanged on the network or those who arrive at a sensitive position and will be transmitted to comparators. Comparators agents with reactive capacity; responsible for comparing the flow of events with the rules and procedures describing the unintended uses.

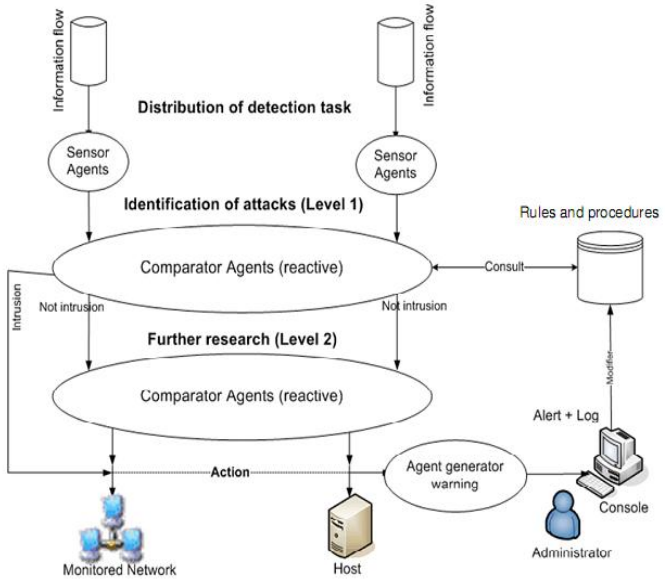

Fig. 6. Proposed platform of intrusion detection.

If a rule is violated when there is interference and the degree of threat that may represent the intrusion, the officer will compare the direct traffic to the cognitive agent to search further, or it blocks traffic and cuts the connection.

Cognitive agents with adaptive and learning function; their role is to check whether the event may represent a low threat and react quickly when an intrusion blocks traffic and prevent the agent generator warning. Agents generating alerts; their role is to generate an alert message to the appropriate administrator and store information about the event in a log file.

4. REALIZATION

In this section, we propose an application [16] realized by the system architecture team of the ENSEM, Hassan II University. In this application, we use the different architectures presented in the previous sections. This realization is an application of the multi-agents system in the remote control and the telecommunication domains.

After the design phase of the proposed model, we have built the first prototype of our solution shown in figure.7. We use soekris box net 5501 to act as a gateway and an open source distribution (Perl) to develop our control architecture for the GSM-based remote wireless automatic monitoring system. The robots used are NXT and Khepera. Concerning the commands send from a cellular phone, they are writing in text message. Once the message is written, we send it. The gateway starts the processes and performs the desired task like the state of the gateway and the bandwidth.

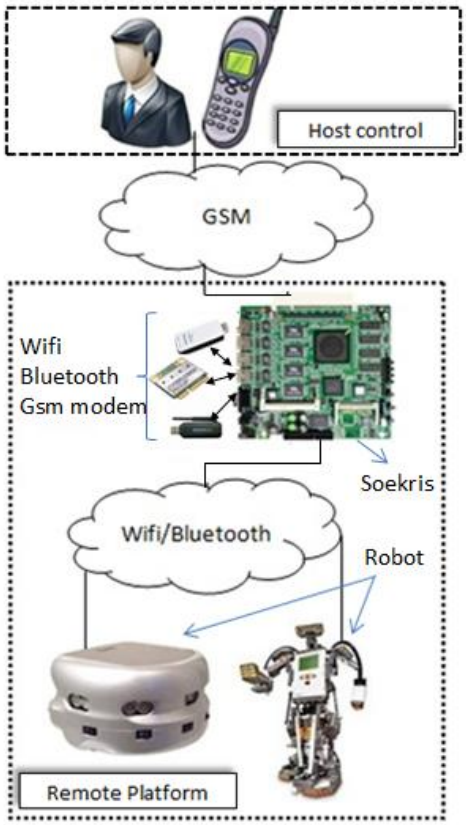

Fig. 7. Prototype of the system

5. CONCLUSION

In this work, we have shown that multi-agents systems applications are numerous. The multi-agents systems can be used for designing and developing distributed, autonomous and intelligent architectures in different fields like remote control, security and telecommunications. The implementations presented in this paper validate the choice of the multi-agents systems.

In future work we hope to increase the intelligence of the agents of our architectures and create a global architecture that allows a securised control on Internet of telecommunications equipments using mobile systems.

6. REFERENCES

[1] Ferber, J. "Multi-agents systems, an introduction to distributed artificial intelligence", Addison-Wesley, 1999.

[2] Medromi, H., Qrichi Aniba, F., Sayouti. A. "Real Time Distributed Architecture based on Multi-Agent System". 2ème Journées d'informatique et de mathématiques décisionnelles (JIMD'2008), ENSIAS, Rabat, Maroc, 3-5 Juillet, 2008.

[3] Benhadou, S., Raoui D., Medromi H. "Nouvelle méthodologie distribuée de sécurité à base de système multiagents", ICeP'09 Marrakech, September 25 – 27, 2009.

[4] Sayouti, A. "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.

[5] Ferber, J. "The multi-agents systems - to collective intelligence", Paris, InterEditions, 1995.

[6] Shoham, Y. "Agent-oriented programming". Journal of Artifcial Intelligence, 60(1), 51-92, 1993.

[7] FIPA 97 specifcation Part 2: Agent communication language, Version 2.0. Foundation for Intelligent Physical Agents. Retrieved from www.fpa.org, 1997.

[8] Odell, J., Paranuk, H. V., Bauer, B. "Extending UML for agents". Proceedings of the AOIS Workshop at the 17th National Conference on Artifcial Intelligence (pp. 3-17), 2000.

[9] David, R., Alla, H. "Petri Nets and Grafcet - Tools for modelling discrete event systems". Prentice Hall, 1992.

[10] Sayouti, A., Medromi, H. "Chapter Title: Autonomous and Intelligent Mobile Systems based on Multi-Agent Systems, Book Title: Multi-Agent Systems - Modeling, Control, Programming, Simulations and Applications", INTECH, http://www.intechweb.org, 2011.

[11] Sayouti, A., Medromi, H. "Book Title: Les Systèmes Multi-Agents : Application au Contrôle sur Internet." Academic Publishing in Europe, August 2012.

[12] Moutaouakkil, F., Medromi, H. "Book Title: Les plateformes de Contrôle autonome distribuée : Application en Robotique Mobile, Academic Publishing in Europe, 2012.

[13] Mansouri, H., Medromi, H., SAYOUTI, A. "A GSM based remote control", The Fourth Workshop on Information Technologies and Communication (Wotic), ENSEM, Casablanca, MOROCCO, 2011.

[14] Benhadou, S., Medromi, H. "Book Title: Sécurité et Détection d'Intrusion à base des Systèmes Multiagent: Nouvelle Approche Distribuée Temps Réel, Academic Publishing in Europe, 2012.

[15] Raoui, D., Benhadou, S., Medromi, H. "New distributed platform for intrusion detection based on multi-agents system", Journal of Engineering and Technology Research Vol.2(10), pp. 200-206, October 2010.

[16] Mansouri, H., Medromi, H., SAYOUTI, A. "A GSM based remote wireless automatic monitoring of mobile Robot", International Journal of Research and Reviews in Computer Science (IJRRCS)", Vol. 3, No. 3, ISSN: 2079-2557, June 2012.