Bulletin of the *Transilvania* University of Braşov • Vol. 3 (52) - 2010
Series I: Engineering Sciences

USING ROBOSMITH FOR MULTIAGENT ROBOTIC SYSTEM

D. FLOROIAN[1] F. MOLDOVEANU[1]

Abstract: *The RoboSmith architecture divide a robot into functional modules such as locomotion, control, sensors, communication, and actuation. Any mobile robot can be constructed by combining these functional modules for a specific application. An embedded software with dynamic task uploading and multi-tasking abilities is developed in order to create better interface between robots and the command centre and among the robots. The dynamic task uploading allows the robots change their behaviors in runtime.*

Key words: *multiagent systems, intelligent agents, modular robots.*

1. Introduction

The main goal of this work is to develop a flexible mini robot to be used in implementation of a multiagent robotic system, and to present the software architecture. The nature of the multiagent systems (even if applied in robotics) brings some limitations and conditions to the design of a reliable platform [1], [5], [11], [13], [14]. Size, cost, and cooperative abilities via specific tools are some of the limitations and conditions. Size and cost limitations are closely related since smaller size means less material thus less cost. With the advances in microelectronics fabrication technologies, the size and cost of the ICs used in the systems went down significantly, which allows the designers to meet the cost limitation. Even though electronic components have gotten smaller significantly, mechanical parts and battery sizes are still reasonably large. Size limitation on a mini robot is imposed by mostly mechanical parts and batteries [3], [6], [9]. The fundamental problem with reducing the size of the mechanical devices is that they either become inefficient or not fully functional when their size is shrunken. For example, when the size of a DC motor or a gear gets smaller, their power and their durability reduce significantly. These limitations have determined our design.

There have been many definitions of a robot in the literature since the beginning of the robotics field in the 1940s. After having been introduced in factories, robots became mobile and smaller with advances made in mechanical and electrical engineering fields. After the mobility of the robots was developed, the artificial intelligence field made its contribution to robotics by making them autonomous and smarter [2], [4], [8], [12].

Multiagent systems represents a relatively new area in computer science and a very

[1] Centre "Control Process Systems", *Transilvania* University of Braşov.

new area in robotics, which started to be developed in 1980s but that only in the mid-1990s gained widespread interest [5], [7], [14], [16]. Multiagent systems are compositions of computing elements that possess autonomous action, and which are able to interact among themselves, not only for exchanging messages but also for a more elaborated kind of communication that resembles social activity (cooperation, coordination, negotiation etc.).

The autonomy of a robot required good communications and sensing skills. These skills can be achieved using multiagent technology, which use agentified components. Also this problem can be approached by using conventional elements for the robot but agentifying the whole robot [1], [12], [16].

Robots need sensors mainly for the reasons. First, sensing is the purpose of the mission. Second, sensing is necessary for survival in the robot's environment: to determine an obstacle on the planned path to the target. Finally, sensing is needed to enable the robots to sense their own configurations and their relationships with the environment. Along with sensing, a robot needs to make decisions (think) to be able to adapt to the environment and to change the environment according to the mission [4], [10], [15].

No machine can survive in an environment without proper feedback from it. Mechanical and sensor errors can easily accumulate and put the robot in a dangerous situation. Thus, cognition is essential for robots' survival. Acting is another essential requirement for the evolution of intelligence. Acting is an ability to act on the environment to survive and accomplish the mission. Manipulation and mobility are key components of action even though they may not be necessary at the same time for small robots. Most of the time, mobility is enough for small robots to accomplish the mission. This changes the current image of a robot from "one-armed iron-laborer" to "a mobile creature" mostly moved by wheels. In recent years, special attention has been given to robots mostly inspired from the nature (e.g. from human cooperation).

Multiagent society is encouraged by human society. Cooperation between lets them accomplish a complex mission with their rather limited skills. This type of behavior brings the communication component to the picture because robots can cooperate with each other only with effective communication between them. A considerable number of papers have been devoted to these topics [1], [5], [10], [12], [14], [16].

The organization of the paper is as follows. A flexible design architecture based on multiagent technology is described in Section 2. Then, in Section 3, an modular mini robot is described following the concepts presented above. In Section 4 is presented the robotic multiagent system, RoboSmith, which link together the mini robots. Finally, Section 5 gives the conclusions.

2. Flexible Design Architecture

There has been a significant amount of research in reconfigurable, modular and flexible robotics in recent years [4], [6], [12]. Most of the research has been on multiple identical modules that construct a single robot. The proposed architecture has a vertical modularity based on horizontal layers multiagent architecture in which the layers are not identical to one another. It slices a robot into functional abstract layers such as locomotion, control, sensors, communication, and actuation.

These concepts avoid main disadvantages of the horizontal layering (which require control of race conditions over the actuators) because there are a single agent which control actuators. Any mobile robot can be constructed by combining the above layers for a specific application. A sub-module is a piece of hardware, which accomplishes the functionality of an abstract functionality

(and also provide that skill for respective agent), i.e. a wireless communication sub-module for a communication layer.

In our flexible architecture, the robot can be constructed by combining a locomotion layer, sensory layers, an actuator layer, and purpose specific layers. The software flexibility is given by multiagent technology. Figure 1 presents the proposed hardware layers and Figure 2 present the software functionality [12].

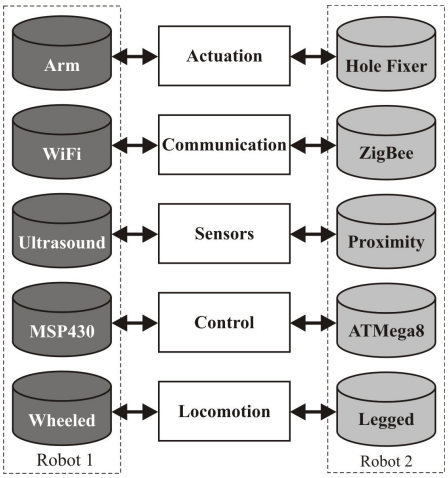

Fig. 1. *Flexible hardware structure*

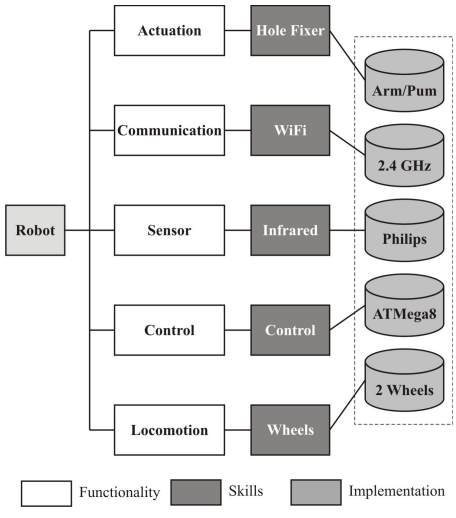

Fig. 2. *Flexible software structure*

In the model, each layer can be implemented by the corresponding hardware. For example, a sensor layer may be an ultrasound sensory board or a proximity sensory board, or perhaps both. The sub-modules are designed such that they have a unique signature and a standard pin connection. The sub-modules can be added at any level and the position does not affect its module's operation.

Since layers can be combined in any order, an application specific robot can be quickly constructed. For example, if a new problem domain requires legs rather than wheels, the wheeled sub-module can be instantly swapped with the legged sub-module. Also, in software application, the wheels agent is replaced with legs agent. This is essential for the flexibility of the applications since agents might be equipped with complementary skills instead of having the same skills.

Programming robotic applications is far from standardized. The primary reason is that each robot is composed of very special hardware designed for a specific goal. The result is that the software also becomes specific. This is very convenient to the multiagent systems which promote reengineering instead of reprogramming. Also the layers architecture can be easily implemented in the multiagent systems. The communications between agents is standardized by FIPA (Foundation of Intelligent Physical Agents) regulations. These facts make multiagent technology very useful in this situation.

A layered architecture is simultaneously reactive and deliberative. The deliberative agents reason and make decisions based on the symbolic representation (model) they have of the external world. These agents need a lot of effort to model the complex entities of the external world. The reactive agents suppose the existence of basic behaviors or sequences of actions that execute concurrently from the lowest level

of intelligence. These behaviors are, in turn, used by more complex ones to create more complex levels of intelligence. A layered architecture contains a set of interacting layers in which some are deliberative and others are reactive. In horizontal layering the sensors are directly connected to each existing layer, which also might drive output directly (see Figure 3).

The focus of this section moves from the architecture of the flexible mini robot to the architecture that a group of agents create a form. Individual agents are useless in the large majority of situations, because most of the scenarios involve several interacting agents. When dealing with multiagent the important aspects are now how they communicate and how they interact. Communication and interaction are the mechanism that let the community of agents has a more complex behavior than just the sum of their individual behaviors.

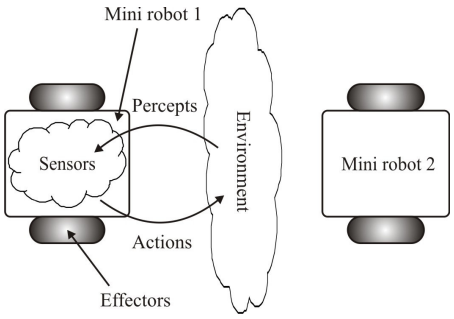

Fig. 3. *Agents interactions with the environment*

For implementation of multiagent systems we will use JADE (Java Agent Development Environment) which is a middleware platform intended for the development of distributed multiagent applications based on peer-to-peer communication [17]. JADE includes Java classes to support the development of application agents and the run-time environment that provides the basic services for agents to execute. An instance of the JADE run-time is called a container, and the set of all containers is called the platform. These platforms provide the layers that hide from agents the complexity of the underlying execution system. This mechanism is depicted in Figure 4.

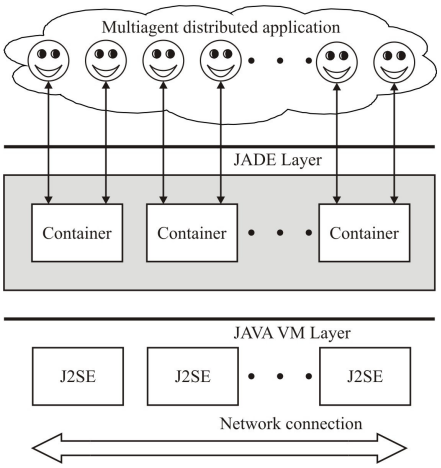

Fig. 4. *The JADE architecture*

3. The Mini Robot

The RoboSmith's robots are flexible mini robots, which take advantage of a layered design approach described in Section II. Even though there are five levels in the hardware and software architectures, the implementations (sub-modules) of the levels can be more than one. In addition, each level may also involve multiple closely related functionalities. The sub-modules are designed and manufactured at Automation Laboratory. This section presents the mini robot and main sub-modules.

Locomotion module has a mechanical base, and locomotion module hardware (sub-module). The base of the robot consists of an aluminium frame, two stepper motors, some gearing, two wheels and associated ball bearings, and the batteries. The base is designed by CAD tools and machined with high precision CNC machines. Figure 5 shows the base with wheels, gears and motors. A legged version

of the base is also in design process as an alternative locomotion to be used in different applications. The battery selected for the mini robot is an AA form factor NiMH rechargeable cell. Four of these cells connected in series are used in the system. The cells are nominally 1.2 volts each for a system voltage of 4.8 volts.

The two wheels module is very versatile and easily directionable. This mechanism allows the robot to make short turns by moving only a wheel and stopping the other. Another advantage is providing by the sensors. The time delay is not critical because, in this situation, the locomotion module has plenty of time to turn and the control module has plenty of time to make decisions.

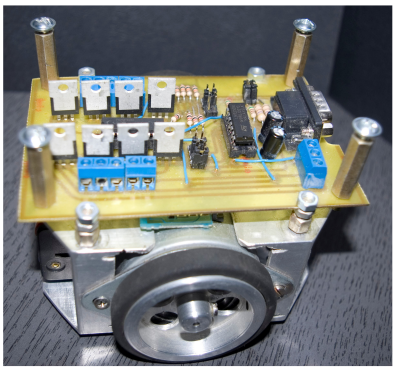

Fig. 5. *The base with wheels for flexible mini robot*

The locomotion layer is implemented by first electronic module and the mechanical base described above (see Figure 6). It's most critical layer in the operation of the robot. It contains the circuit for the stepper controller, which provides direction control for the motors and supplies the high current they require. This module is also manufactured in laboratory and includes common components. It also includes the power system, which consists of a DC-DC converter and some passive components. The power system provides +5 volts for the entire robot, and will accept from 1.5-15 volts on its input. This provides plenty of flexibility if a different battery system is put in place. This sub-module also contains the charge circuit, which allows the battery to remain in the robot while it's being recharged. Also by choosing high capacity elements the robot has more autonomy.

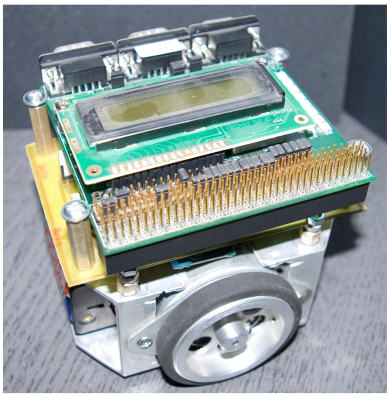

Fig. 6. *Implementation of locomotion layer of flexible mini robot*

Over on this sub-module (see Figure 7), is the main controller, an ATMega8 microcontroller running at 16 MHz. The 8 kB flash memory is included on chip and also 512 B RAM and 1 kB SRAM. All other components are soldered directly to the board. With improved memory architecture, the mini robots are able to run in a flexible architecture.

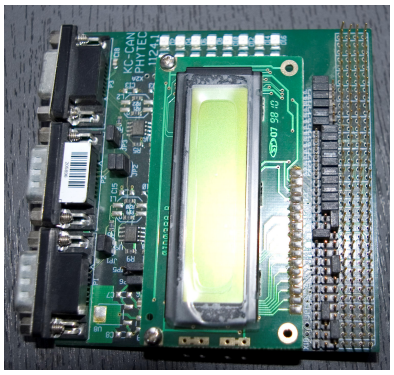

Fig. 7. *Mini robot's CPU*

At the top, the sub-module for communication layer is based on a XBee hardware board (serial version) (which is very similar to familiar ZigBee modules) [19]. This is necessary for agent's interactions and for reporting to the main server unit which is hosted on a PC (see Figure 8). Another XBee module (an USB version) is connected to a host PC and connects the mini robot to the control program. Also many mini robots (a maximum of 16 is recommended) equipped with XBee module can be interconnected in this manner. The control program will coordinate the messages.

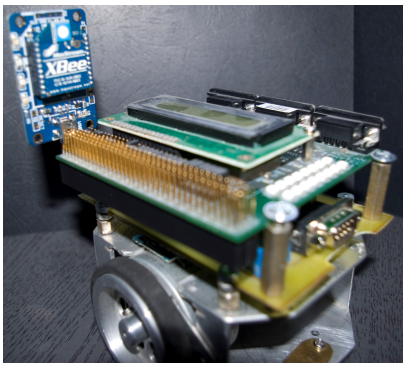

Fig. 8. *Implementation of hardware communication layer*

4. The Multiagent Robotic System

The RoboSmith architecture is based on a multiagent robotic system which coordinates the entire community of agents. This section presents the implementation of RoboSmith architecture. The RoboSmith is a networked organization of mini robots, which have together formed a cooperative dynamic network to reach group benefits. RoboSmith is a society of agents (the mini robots) and therefore their interactions are at society level. The mini robots can be considered intelligent agents because they are proactive, reactive and have social ability. They are proactive since they have goal directed behavior that is seen when the layers involved participate in society with the best possible performance. Moreover, they can keep working even under environmental coalitions. They are reactive in the sense that they react to changes in the external environment, which are "sensed" through messages. Although RoboSmith agents are not purely reactive, the importance of messages in their behavior is so relevant that their architecture is more reactive than proactive. Finally, they have social ability because they are able to negotiate and cooperate with the other mini robots.

Communication and interaction among individuals and about a domain can only take place if some conceptualisation of that domain exists. To guarantee a common semantic understanding, agents must use an appropriate ontology to communicate with their partners. In the case of RoboSmith, all the mini robots need to share some basic concepts, such as skills, requests, services and agent. Therefore all agents of the proposed architecture share a basic global ontology that models the basic referred concepts.

For our purposes, we have adopted the description of an agent as a software program with the capabilities of sensing, computing, and networking associated with the specific skills of the mini robots described above. This implementation is made in JADE because this development tools is very versatile and could be very well integrated with others development tools (like Protégé-2000 and Java [17], [18]). Also JADE is an open source FIPA compliant Java based software framework for the implementation of multiagent systems. It simplifies the implementation of agent communities by offering runtime and agent programming libraries, as well as tools to manage platform execution and monitoring and debugging activities. These supporting tools are themselves FIPA agents.

JADE offers simultaneously middleware for FIPA compliant multiagent systems,

supporting application agents whenever they need to exploit some feature covered by the FIPA standard (message passing, agent life cycle etc.), and a Java framework for developing FIPA compliant agent applications, making FIPA standard assets available to the programmer through Java object-oriented abstractions. The general management console for a JADE agent platform (RMA - Remote Monitoring Agent), like in Figure 9, acquires the information about the platform and executes the GUI (Graphic User Interface) commands to modify the status of the platform (creating new agents, shutting down containers etc.) through the AMS (Agent Management System).The DF (Directory Facilitator), AMS, and RMA agents coexist under the same container (main-container) together with the RoboSmith's agentified mini robots, as it is shown in Figure 9.

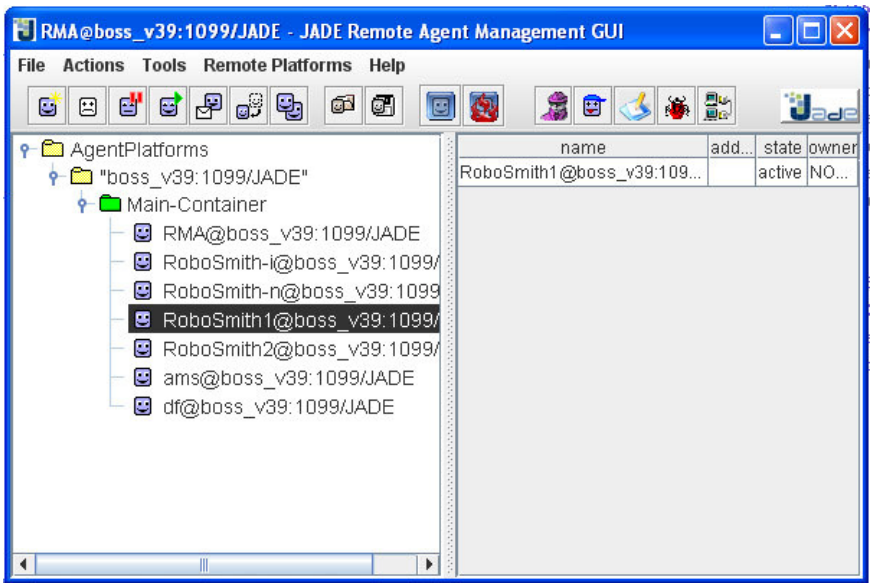

Fig. 9. *JADE Implementation of RoboSmith*

To facilitate message reply, which, according to FIPA, must be formed taking into account a set of well-formed rules such as setting the appropriate value for the attributes in-reply-to, using the same conversation-id etc., the method createReply() is defined in the class that defines the ACL (Agent Communication Language) message. Different types of primitives are also included to facilitate the implementation of content languages other than SL (Standard Languages), which is the default content language defined by FIPA for ACL messages. This facility is made with Protégé 2000 as depicted in Figure 10.

In Figure 11 is presented a graphical view of ontology classes which facilitate understanding the fact that from the point of view of the programmer, a JADE agent is simply a Java class that extends the base agent class. It allows inheriting a basic hidden behavior (such as registration, configuration, remote management etc.), and a basic set of methods that can be called to implement the application tasks of the agent (i.e. send/receive ACL messages).

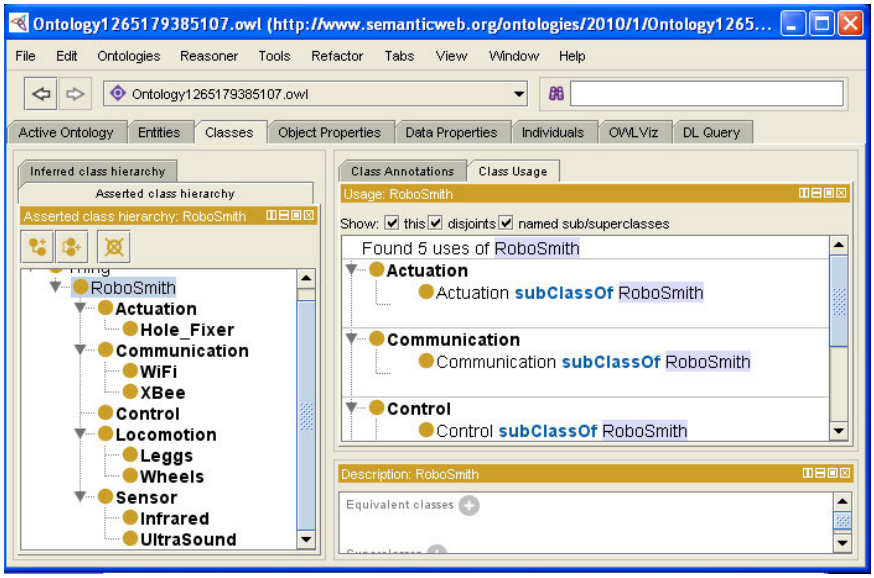

Fig. 10. *Defining ontologies for RoboSmith*

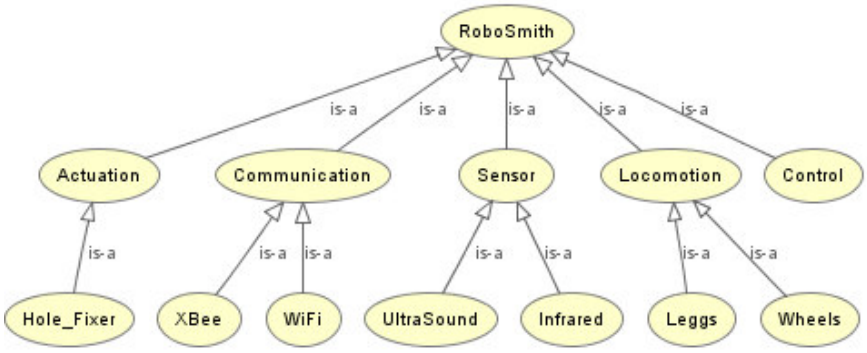

Fig. 11. *Classes hierarchy for RoboSmith architecture*

Moreover, user agents inherit from their Agent superclass some methods to manage agent behaviors. Also this diagram can be represented in UML (Unified Modelling Language) because behaviors are implemented as hierarchy of classes. The Protégé 2000 connects to RoboSmith JADE Agents by including a Protégé configuration file in the Java compiler. The RoboSmith ontology is divided in many concepts that follows the class hierarchy defined above.

5. Conclusions

In this paper, the exploitation of multiagent technology applied in flexible mini robotic system has been presented. A number of two robots were constructed and implied in multiagent robotic system It is possible to extend this number by adding similar robots. The proper function of the robotic system is important from a practical point of view since it provides a

detailed framework about the design of the control structure and the behaviors tasks. This confirm that multiagent technology can be successfully implemented for control the robotic systems.

The RoboSmith architecture is a multiagent system which could coordinates many mini robots in order to achieve their goals. In the same time, RoboSmith could dynamically change the behaviour of the robots, in real time, by rearranging the order and the task of each layer.

References

1. Barata, J., Camarinha-Matos, L.M.: *Multiagent Coalitions of Manufacturing Components for Shop Floor Agility - The CoBaSA Architecture.* In: International Journal of Networking and Virtual Organisation **2** (2003) No. 1, p. 50-77.
2. Choi, B.S., Lee, J.J.: *Localization for a Mobile Robot Based On an Ultrasonic Sensor Using Dynamic Obstacles.* In: Artificial Life and Robotics **12** (2008) No. 1-2, Springer Japan, p. 280-283.
3. Denneberg, V., Fromm, P.: *OSCAR. An Open Software Concept for Autonomous Robots.* In: Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society, Aachen - Germany, August 31 - September 4, 1998, p. 1192-1197.
4. Feng, L., Borenstein, J., et al.: *Where am I? Sensors and Methods for Autonomous Mobile Robot Positioning*. *Vol. 3*. The University of Michigan, 1994, p. 9-10.
5. Floroian, D.: *Sisteme Multiagent* (*Multiagent Systems*). Cluj-Napoca. Editura Albastră, 2009.
6. Jueyao, W., Xiaorui, Z., et al.: *Design of a Modular Robotic System for Archaeological Exploration*. In: International Conference on Robotics and Automation, ICRA'09, Kobe, Japan, 12-17 May 2009, p. 1435-1440.
7. Komatsu, T., Kuki, N.: *Investigating the Contributing Factors to Make Users React toward an On-screen Agent as if They are Reacting toward a Robotic Agent*. In: 18th IEEE International Symposium on Robot and Human Interactive Communication, Toyama, Japan, September 2009, p. 651-656.
8. Lee, D., Chung, W.: *Discrete-status-based Localization for Indoor Service Robots*. In: IEEE Transaction on Industrial Electronics **53** (2006) No. 5, p. 1737-1746.
9. Lee, W.H., Sanderson, A.C.: *Dynamics and Distributed Control of Modular Robotic Systems.* In: Proceedings of the 26th Annual Conference of the IEEE Industrial Electronics Society, Nagoya, Japan, 22-28 October 2000, p. 2479-2484.
10. Opp, W.J., Sahin, F.: *An Artificial Immune System approach to Mobile Sensor Networks and Mine Detection.* In: IEEE SMC 2004, Hauge, The Netherlands, October 2004, p. 947-952.
11. Petriu, E., Whalen, T., et al.: *Robotic Sensor Agents: A New Generation of Intelligent Agents for Complex Environment Monitoring.* In: IEEE Instrumentation & Measurement Magazine **7** (2004) No. 3, p. 46-51.
12. Sahin, F.: *GroundScouts: Architecture for a Modular Micro Robotic Platform for Swarm Intelligence and Cooperative Robotics*. In: International Conference of Systems, Man and Cybernetics (2004), p. 929-934.
13. Weyns, D., Parunak, H.V.D., et al.: *Environments for Multiagent Systems State-of-the-Art and Research Challenges*. In: E4MAS'04, Springer-Verlag, Berlin, 2005, p. 1-47.
14. Wooldridge, M., Jennings, N.R.: *Agent Theories, Architectures, and Languages: A Survey*. In: Proceedings of ECAI'94 -

Workshop on Agent Theories Architectures and Languages, Amsterdam, The Netherlands, August 1994, p. 145-160.
15. Zheng, J., Lorenz, P., et al.: *Guest Editorial: Wireless Sensor Networking*. In: IEEE Network **20** (2006) No. 3, p. 4-5.
16. Suboting, S.A., Oleinik, A.: *Multiagent Optimization Based on the Bee-Colony Method.* In: Cybernetics and Systems Analysis **45** (2009) No. 2, p. 177-186.
17. *** Java Agent Development Framework - JADE. Available at: http://jade.tilab.com/. Accessed: 12-02-2010.
18. *** Protégé 2000. Available at: http://protege.stanford.edu/. Accessed: 12-02-2010.
19. *** XBee. Available at: http://www.digi.com. Accessed: 26-01-2010.