Chidori – a bio-inspired cognitive architecture for collective robotics applications

Cristian Vasile, Ana Pavel, Cătălin Buiu

Laboratory of Natural Computing and Robotics, Department of Automatic Control and Systems Engineering, "Politehnica" University of Bucharest, Spl. Independentei 313, 060042 Bucharest, Romania, (e-mail: {cvasile, apavel, cbuiu}@ics.pub.ro).

Abstract: A cognitive collaborative multi-agent control architecture that addresses real-world control problems for swarms of mobile robots is proposed. The swarm's emergent behaviour is obtained by using a distributed Particle Swarm Optimization inspired algorithm. A swarm-user interface is also presented and offers a way for a human operator to interact with and guide the robotic swarm without limiting its emergent intelligence. The architecture is designed as a multi-agent system developed using JADE Framework. Three types of agents were defined: local behaviour agent, social behaviour agent and graphical user interface (GUI) agent. Each robot is associated with a pair of a local and a social behaviour agents which implement the reactive component and the interaction between the robots. The swarm forms a hierarchical structure composed of subswarms and neighbourhoods based on tasks and goals defined by the user or the swarm itself. The GUI agent is used as a link between the human expert and the swarm. The architecture was tested on a swarm of e-puck robots.

Keywords: bio-inspired cognitive architecture, collaborative robotics, swarm intelligence, swarm-user interface, multi-agent system, particle swarm optimization.

1. INTRODUCTION

There are many problems that need to be solved by a robotic system in real-world environments, which are often dynamic, unstructured and unpredictable. One research focus is the development of cognitive architectures for collaborative multi-robot systems. These may offer advantages such as fault-tolerance, scalability, flexibility and robustness which make them better than single robot systems. One approach is to use bio-inspired techniques to improve the performance of a multi-robot system. The Swarm Intelligence (SI) paradigm uses multiple simple autonomous agents and it is inspired by the behaviour of bird flocks, fish schools and insect societies. Robustness of a swarm system is given by its high number of agents. A task may be successfully completed even if some of the robots are lost or are not working properly. SI based systems have distributed control and information is used by every individual to update its state or react to it. There is no single decision-making module, and the system's goal oriented emergent behaviour is given by the interaction between the robots and between the robots and the environment (Martinoli, 1999).

This paper proposes a cognitive collaborative multi-agent control architecture, which was named Chidori (lit. "One Thousand Birds", in Japanese, to underline its dedication to control swarm systems). Knowledge is used in the process of social interaction between robots. The swarm system is capable of learning even though the robots which compose it do not have such capabilities. The architecture is implemented as a multi-agent system. This approach is natural because software agents are mapped directly over the robots' functionalities. A detailed description will be given in section 3.

Based on the proposed architecture a swarm robotic system was implemented in order to accomplish tasks like finding a target in an unknown and hazardous environment, transporting dangerous or toxic materials and assist in rescue missions. Some specific scenarios are detailed in section 2. Most of these tasks employ some form of intelligent search methods. The proposed system uses the Particle Swarm Optimization (PSO) technique. The PSO method is a global stochastic search technique which uses individuals called particles to find the global optimum of a fitness function. The particles fly through the problem search-space according to their local optimum and the global one (Kennedy and Eberhart, 2001). The PSO technique is very powerful, but in order to use it with multi-robot systems, some modifications must be made. Some issues concerning the implementation of a PSO based algorithm on a swarm are presented in Pugh and Martinoli (2007a, 2008). They also show that an adapted PSO-based algorithm can find a target in a simulated environment. In another study, PSO has been proven to have better performance than a Genetic Algorithms approach (Pugh and Martinoli, 2006). Hybrid PSO with fuzzy logic controllers strategies have also been considered (Venayagamoorthy et al., 2008). The effects of real-world factors (Pugh and Martinoli, 2007b) and group size (Hayes, 2002) in a PSO-inspired algorithm were also studied.

The proposed architecture is designed to offer an interface between a human operator and a swarm. This aspect is very important because an operator must be able to guide a swarm to accomplish its goals without limiting the swarm's emergent intelligence. The operator, as an expert, may have knowledge about the state of the environment in which the tasks are performed. Because the number of robots in a swarm may be very large, the operator may not and should not set goals (supervise) for each robot. Instead he should be able to set goals for the whole swarm and let it self-organize in order to accomplish them. Thus, the operator does not control the swarm, but guide it. Swarm-user interfaces are needed in order to solve problems in real situations where humans and robots must work together.

2. PROBLEM STATEMENT AND SCENARIOS

The proposed control architecture addresses the problem of developing multi-robot systems deployable in search missions with wide search-space and in various dangerous situations such as calamities or industrial accidents. The proposed architecture must be scalable, flexible, robust, extensible, distributed and must also allow the interaction with human experts. Several scenarios have been imagined in order to develop a general architecture for real-world applications. Some of these scenarios are detailed below.

Scenario 1: Intervention in case of an accident. An accident occurred in a factory that uses toxic materials. The space becomes dangerous to humans and should be cleaned urgently to not contaminate the environment. The area where the accident occurred does not allow the use of large equipment. A human operator will guide the intervention robotic colony to remedy the situation. They will search and transport the toxic materials and will store them in special containers.

Scenario 2: Searching for leakage of toxic substances. In an industrial space equipped with special sensors, leakages of toxic substances (CO2, NH3, CH4, industrial detergents, etc.) have been detected. The space cannot be accessed by people without special equipments. To reduce the risk of accidents (with human victims) an intervention robotic team is prepared to act. The swarm will be guided by a human operator and it will determine the source of leakages. According to the information that may be available, the human operator can enhance the search effort of the swarm. A swarm would be useful in such situations because it could search in parallel, using the human operator's suggestions and the stochastic search algorithm.

Scenario 3: Hazardous materials transportation. In a factory, it is necessary to transport and storage some toxic materials in a particular area. The intervention robotic team is used for this operation. Guided by the human operator, which defines the transportation parameters (trajectory, velocity, phases etc.), the team will carry on the loads safely. A robotic swarm has the advantage of being capable of transporting loads that cannot be handled by a dedicated machine. Also, it may be useful in situations in which transportation is not performed very often and the purchase of a dedicated machine would not be profitable.

3. THE PROPOSED SWARM ARCHITECTURE AND SWARM-USER INTERFACE

3.1 Overview of Chidori

The proposed cognitive collaborative multi-agent control architecture has both reactive and hierarchical components. The multi-robot system has distributed control and distributed perception, but the cognition is given by the emergent behaviour of the swarm. Each robot has its own set of sensors to perceive the world and react to it based on its own internal state and its social interaction with the other robots.

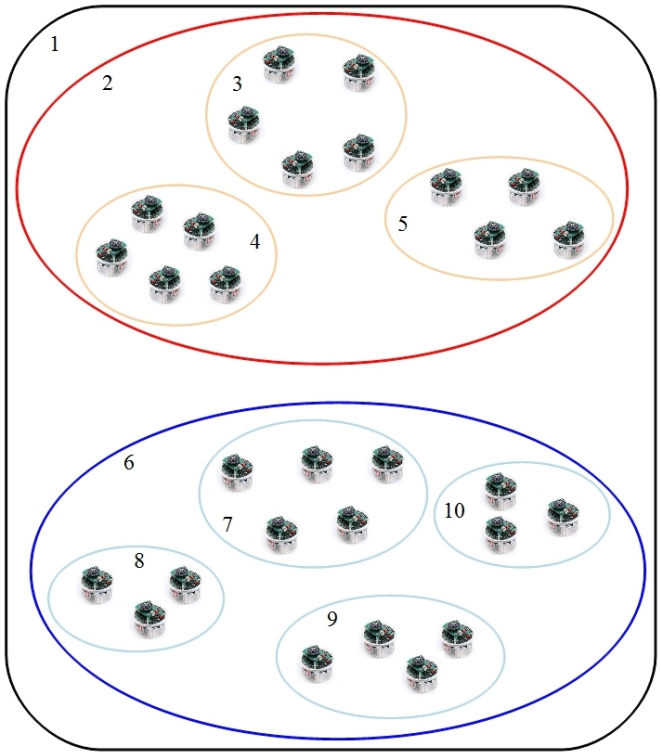

Fig. 1 The hierarchical structure of the architecture – the outer border (1) encloses the whole swarm in which two subswarms have formed, delimited by borders 2 and 6. In each subswarm a number of neighbourhoods can be noticed (3, 4, 5 in one subswarm and 7, 8, 9, 10 in the other subswarm)

The hierarchical component (Fig. 1) is created from the social interaction between the robots. Subswarms (group of robots) are formed inside the swarm in order to accomplish specific tasks. Neighbourhoods (smaller group of robots) can be formed inside a subswarm in order to help each other. Both subswarms and neighbourhoods can be created or modified in a static or dynamic way. Subswarms are formed when the swarm needs to accomplish multiple different goals, like searching for different targets. On the other hand, neighbourhoods are formed in order to accomplish their common goal and to improve the solving capabilities of a subswarm.

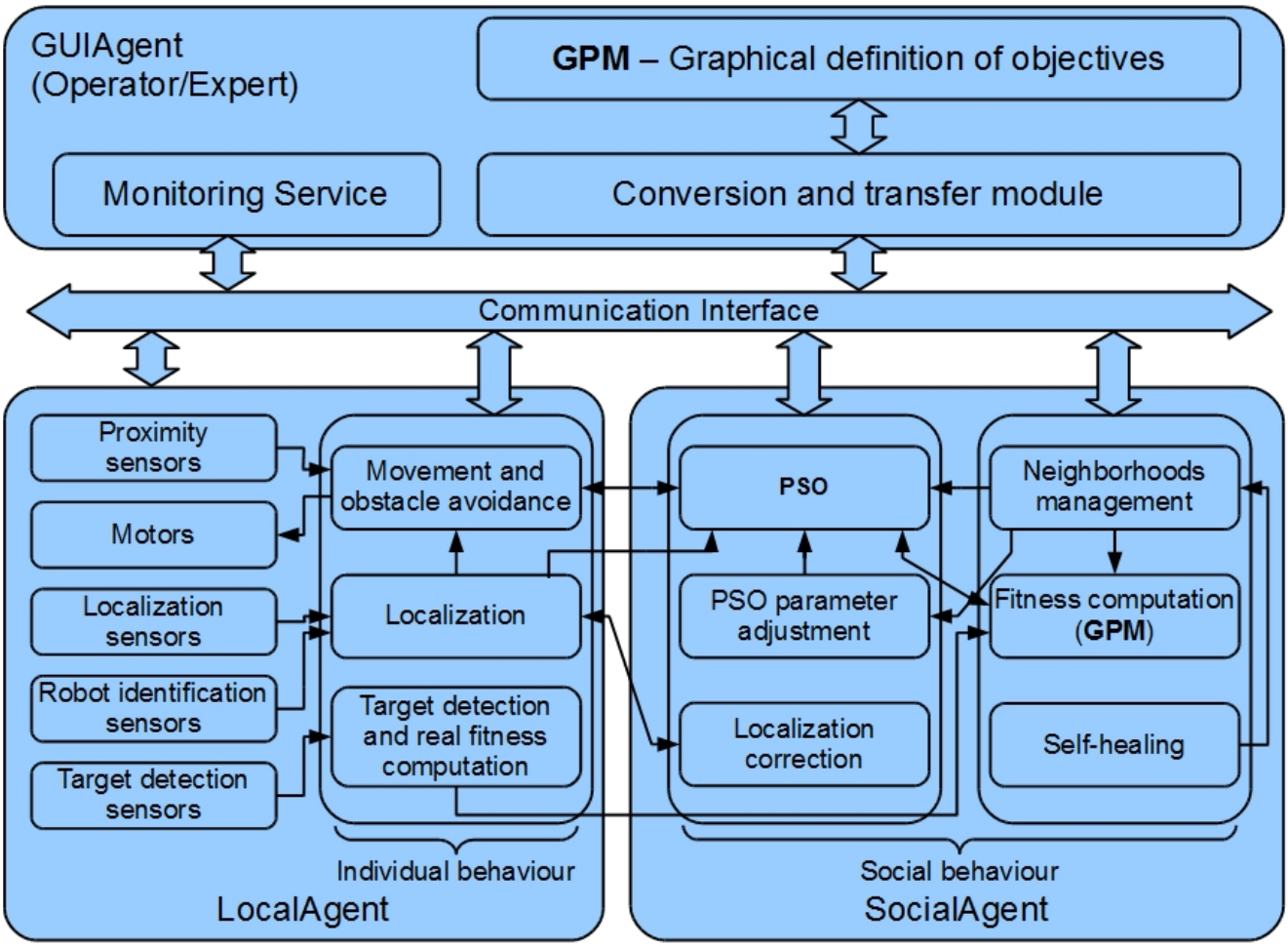

Fig. 2 Chidori Architecture

The architecture is implemented as a multi-agent system (Fig. 2). Each robot is associated with two agents, a local behaviour agent and a social behaviour agent.

The swarm-user interface is based on Gravity Points Method (GPM) (Vasile et al., 2010) and is implemented both in the interface agent, GUIAgent, and in the Fitness Computation Module in the Social Behaviour Agent.

3.2 The LocalAgent

The local agent is responsible for the robot's individual behaviour. It interacts directly with the hardware, sometimes through the operating system or other middleware, and it is, thus, dependent on the type of robot it runs on. The local agent interacts with the social agent in order to act pro-actively based on the group's (neighbourhood, subswarm, swarm) goals.

The local agent can be associated with a real robot or simulated robot, allowing quick testing and debugging of the system.

The functionalities it provides are implemented as behaviours of the agent (see section 4). There is a special update behaviour, a ticker behaviour, that is used to send commands and read sensors' values. The local agent's modules use the perceived state of the robot saved in the agent's memory. The basic hardware components that the local agent needs are: proximity sensors (IR on e-puck) used for the estimation of distances to obstacles, motors, localization (encoders on the motors) sensors used for position estimation, robot identification sensors (IR on e-puck) used to detect and identify another robot and target detection sensors (microphones on e-puck).

The local agent was designed to contain three basic modules that need to be implemented:

Movement and obstacle avoidance module. This module receives commands from the social agent and it is responsible for moving the robot to the given position based on the information received from the localization module. If the robot encounters an obstacle, it will try to avoid it and then it will wait for a new command.

Localization module. It uses odometry, the encoders on the motors, to compute the robot's position. Because this method accumulates errors over time, the module collects data about encounters with other robots and sends the data to the social agent's localization correction module. This method is used instead of a global positioning one in order to allow the system to be used in areas where such a system is not

available, like outer space or underground. A hybrid localization system is also desired, when available, because the local positioning is very accurate on short distances and has a high update rate and the global positioning can be used to nullify the error, but with a much lower sampling rate.

Target detection and real fitness computation module. This module is very important in search tasks, because it is responsible for the target's detection. In most search tasks, the target can't be perceived directly because a robot has to enter the target's detection range. This constrain is determined by external factors such as noise and signal strength for radiation sources or wind strength and direction for odour sources. This module is also responsible for computing a fitness value that is sent to the social agent to be used by the other robots in order to find the real target.

3.3 The SocialAgent

The social agent is that component of the architecture which generates the swarm's collaborative and emergent behaviour. The agent is responsible for organizing, managing and controlling the swarm of robots in a distributed, scalable way. The modules belonging to the social agent form two classes: the low-level and high-level behaviour modules.

The high-level behaviour modules interact with the operator through the GUIAgent in order to accomplish the given tasks. These modules define the structure and the goals of the swarm.

PSO module. The PSO module is the control module of the robot. It uses the fitness function given by the Fitness Computation module and the social position in the swarm hierarchy (subswarm and neighbourhood) to compute the next position for the robot to move to. The module uses an adapted, distributed PSO-inspired algorithm to control the robot's movements. The algorithm's parameters can be externally adjusted by the dedicated module.

PSO parameter adjustment module. The parameters of the PSO-inspired algorithm can be adjusted in an adaptive dynamic or static way in order to fine tune the swarm's behaviour. The adjustment is done in accordance to the neighbourhood's goals and can vary between individuals, even from the same neighbourhood.

Localization correction module. Position of the associated robot is corrected based on the social interaction with the encountered robots. Localization data from all the robots involved in the encounter is used to negotiate correction values for their positions. Information such as current position, number of previous encounters and computed correction values can be used in the negotiation phase. The method used to compute the final positions needs to ensure that, given enough encounters, the positioning error is maintained below certain bounds over the operation time of the swarm. It is important to note that this module does not require for the robots to be part of the same neighbourhood or even the same subswarm.

Neighbourhoods management module. This module is responsible for allocating robots to subswarms and neighbourhoods based on the tasks and goals given by the human operator or the swarm. It, thus, creates and dynamically modifies the hierarchical structure of the swarm in order to maintain a balanced distribution of the swarm's robots. The module does a spatial (parallel) planning. Temporal planning (scheduling) of the tasks and goals is done in the GUIAgent. As illustrated in Fig. 1 the hierarchical structure of the swarm can be mapped onto a P membrane system (Paun and Paun, 2004).

Fitness computation module. The fitness values computed by this module are used by the PSO module to control the robot. The values are computed based on the Gravity Points Method (GPM), which uses the operator's and the swarm's own goals in the evaluation. The goals are expressed, according to the GPM method, as virtual attractive or repulsive points, and are used to estimate the position of the target or to highlight dangerous areas, respectively. The module also takes in consideration events of real target detection. In such cases it will use the real fitness value, instead of the virtual one based on virtual attractive points, to compute the final value.

Self-healing module. The swarm's own goals are formed in the self-healing module, so that the robots help each other. A robot that is stuck in a hole or needs help to overcome an obstacle may use this mechanism to generate goals in order to make the other robots aware of it, so that they try to come and help it. This mechanism is very important, because it gives the swarm a superior level of autonomy.

3.4 The GUIAgent

The GUIAgent is the interface of the human operator to the swarm. It is used to define the tasks and parameters which the swarm must accomplish. With this interface the expert operator can guide the swarm to successfully complete its missions.

In order to achieve the desired results, three modules were included in the design: a module to graphically define the goals of the swarm based on the Gravity Points Method (GPM), a module to convert and transfer the goals to the swarm according to a temporal plan (schedule) and a monitoring service, which provides robot-level, neighbourhood-level, subswarm-level and swarm-level information.

4. INTEGRATED TECHNOLOGIES

The developed cognitive multi-agent architecture is based on JADE framework which is a Java based technology. As illustrated in Fig. 3, the multi-agent system is designed based on a number of different technologies working together. The project is implemented using the OOP Java language and five integrated technologies: Protégé, JADE, JavaFX, Webots Simulator and e-puck autonomous robot.

These technologies are linked together as follows. Using JADE platform, different types of agents were implemented.

One of these agents is the GUIAgent which is responsible for the interaction with the operator. JavaFX technology has been chosen to build the graphical component of this agent because it can be easily integrated with Java based applications.

The professional robotics simulator Webots is used for testing and debugging the multi-agent application. The JADE local behaviour agents are interfaced with the simulated robots by TCP/IP. This interfacing method is the easiest way of integrating a third-party program with Webots simulator. However, the simulator has a major advantage which is the option of connecting and transmitting commands from the graphical interface, directly to the real robots. Thus, it is not necessary to modify the code when working on real robots.

Another important technology integrated in the project is the Protégé platform. Using Protégé, the ontologies used by the agents to communicate are defined. The ontologies are exported from Protégé using the OntologyBeanGenerator plug-in which converts them into FIPA/JADE compatible ontology Java classes.

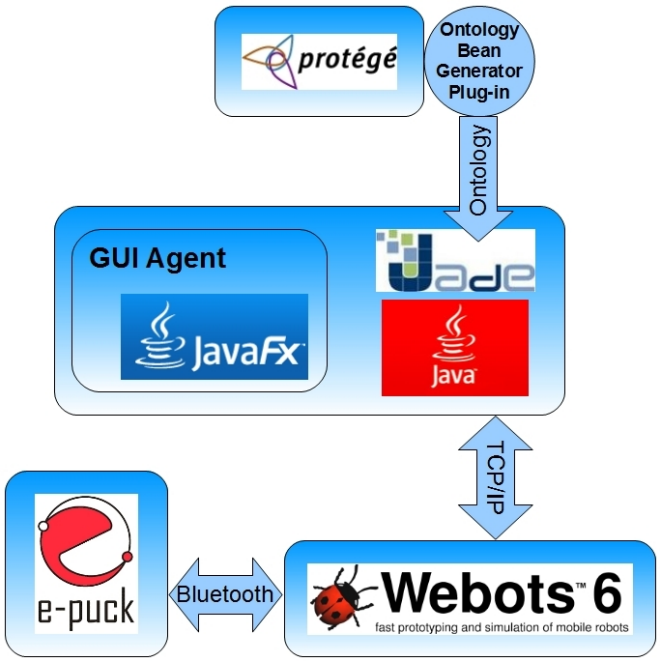

Fig. 3 Integrated technologies used in the Chidori architecture

Some important details regarding the above mentioned integrated technologies will be specified in the following subsections.

4.1 JADE

JADE is a middleware Java based software framework which offers platform independent facilities for distributed applications. Some important features of JADE are:

- JADE is compatible with FIPA specifications;
- it offers support for agents' mobility;
- it allows implementation of white pages and yellow pages;
- it offers support for Java2MicroEdition;
- it offers support for ontologies and semantic languages (SL, Leap, XML) for content message representation;
- it allows defining different agent behaviours like: simple behaviour, cyclic behaviour, ticker behaviour;
- the conversion between Java objects which are internally used by JADE and the platform independent messages represented through a semantic language is performed by the Content Manager class. Content Manager class allows selecting a codec which converts Java objects into a semantic language representation and reverse (Fig. 4). Semantic languages are standard, platform-independent languages used for communication between agents situated on the same or on different machines or platforms.

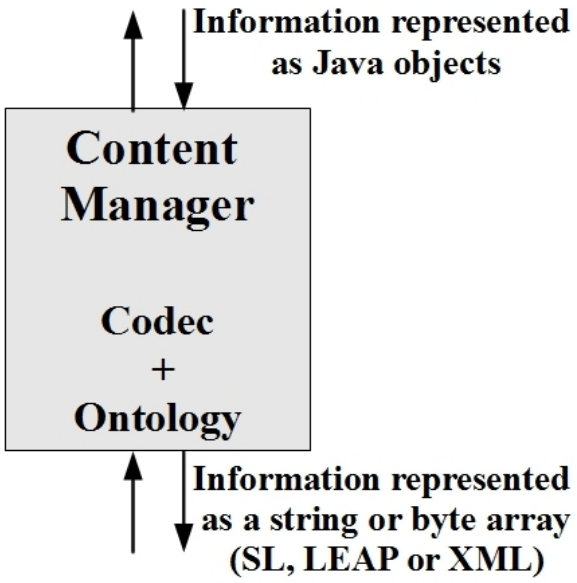

Fig. 4 JADE ContentManager

More details about JADE technology can be found in (Bellifemine et al., 2007).

4.2 Webots

Webots is a simulator for robotics, used in academic and research areas. It is used for testing, debugging and simulating the application before porting it on real robots. Among the properties and parameters which can be simulated with Webots are velocity, inertia, friction, material properties based on specific spring and damping constants. Webots also offers a various set of models for different types of robots, sensors and actuators.

4.3 JavaFX

JavaFX is a platform dedicated for building and delivering rich multimedia applications which run on different devices. The associated language is JavaFX Script declarative

language which is compiled in Java byte code. JavaFX applications run on any platform with a JRE virtual machine (desktop or embedded systems with JavaME). JavaFX scripts are short, easy to use, test and debug.

This software technology fits very well with the proposed Java based application and it is used for developing the graphical user interface of the GUIAgent.

4.4 Protégé

Protégé is an open source framework which offers various tools for creating domain-specific ontologies. The ontology is the common vocabulary understood by all the agents in the system and it is a knowledge representation method. Some examples of defined concepts are: RobotModel, Sensor (IR, US), Actuator (MotorDC, Led), Position. Different predicates (like: HasPosition, etc.) and different actions (like: MoveToPosition, etc.) are also defined in the ontology.

4.5 e-puck autonomous robot platform

e-puck is a small, autonomous mobile robot with two differential wheels (Fig. 5). It is designed to be open hardware and its software is open source. A number of e-puck robots (5-10) is used for testing the proposed architecture as this type of robot is suitable for swarm robotic applications because it is small, easy to use and manipulate. The robot's important features for the proposed multi-robot system are: embedded dsPIC30CPU@30Mhz, 2 DC motors, 8 infrared proximity sensors, 3 microphones, one loudspeaker, 2 hours functioning autonomy, 13 cm/s speed. The IR sensors are used for obstacle avoidance and for detecting other robots. In the testing application, the target which must be found by the swarm is a sound source which is detected by the robots through the microphones.

Fig. 5 e-puck robots in a collective robotics application

Thus, e-puck robot is used for testing the proposed architecture in real world environments before it is implemented for industrial or other human-swarm co-worker applications for solving real world problems.

5. CONCLUSIONS

A multi-agent implementation for the proposed cognitive collaborative control architecture was presented. Techniques, technologies and design aspects have been discussed. Also, the importance of a swarm-user interface and relevant design issues has been pointed out. Preliminary experiments of the proposed multi-robot system were successfully conducted. Future work will include further testing, fine tuning of the local and social behaviour agents and various user-cases will be conducted to evaluate the swarm-user interface performance.

6. ACKNOWLEDGEMENT

This work was supported by CNCSIS – UEFISCSU, project number PNII – IDEI 1692/2008. The contribution of Octavian Arsene to the initial design of the architecture is acknowledged.

REFERENCES

Bellifemine F., Caire, G., and Greenwood, D., (2007). *Developing Multi-Agent Systems with JADE.* Wiley, West Sussex, England.

Eberhart, R., Shi, Y., and Kennedy, J., (2001) *Swarm intelligence.* Morgan Kaufmann, USA.

Hayes, A., (2002). *How Many Robots? Group Size and Effciency in Collective Search Tasks.* Proc. of the 6th Int. Symp. on Distributed Autonomous Robotic Systems, Fukuoka, Japan, 289–298.

Paun, G. and Paun, R., (2004). *Membrane Computing and Economics: Numerical P Systems.* Fundamenta Informaticae, 213–227.

Pugh, J. and Martinoli, A., (2006). *Multi-Robot Learning with Particle Swarm Optimization.* Int. Conf. on Autonomous Agents and Multi-agent Systems, 441–448.

Pugh, J. and Martinoli, A., (2007). *Inspiring and Modeling Multi-Robot Search with Particle Swarm Optimization.* IEEE Swarm Intelligence Symposium, 332–339.

Pugh, J. and Martinoli, A., (2007). *The Cost of Reality: Effects of Real-World Factors on Multi-Robot Search.* IEEE Int. Conf. on Robotics and Automation, 397–404.

Pugh, J. and Martinoli, A., (2008). *Distributed Adaptation in Multi-Robot Search using Particle Swarm Optimization.* 10th Int. Conf. on the Simulation of Adaptive Behavior. Lecture Notes in Computer Science, 393–402.

Martinoli, A., (1999). Ph.D. Thesis: *Swarm Intelligence in Autonomous Collective Robotics: From Tools to the Analysis and Synthesis of Distributed Control Strategies.* Lausanne, Switzerland, EPFL.

Vasile, C., Pavel, A., and Buiu, C., (2010). *Design and implementation of an innovative swarm control architecture*, submitted.

Venayagamoorthy, G., Grant, L. and Doctor, S., (2008). *Collective robotic search using hybrid techniques: Fuzzy logic and swarm intelligence inspired by nature.* Engineering Applications of Artificial Intelligence, 431–441.