AQTR 2004 (THETA 14)
2004 IEEE-TTTC – International Conference on Automation, Quality and Testing, Robotics
May 13 - 15, 2004, Cluj-Napoca, Romania

ROBOTUX, A MULTIAGENT ROBOT BASED SECURITY SYSTEM

Radu Bogdan Rusu[1]

Supervisor(s): prof. Liviu Miclea[2]**, assist. Szilard Enyedi**[3]

Technical University of Cluj-Napoca,
Faculty of Automation and Computer Science,
Department of Automation and Industrial Informatics
26-28 Baritiu Str., RO-400027, Cluj-Napoca, Romania
Tel.: +40-264-594469
E-mail: [1]*Radu.Rusu@aut.utcluj.ro*, [2]*Liviu.Miclea@aut.utcluj.ro*, [3]*Szilard.Enyedi@aut.utcluj.ro*

Abstract:

This paper presents an alternative approach to a classical security system. Our solution, called Robotux employs mobile robots and software Java agents to create a more efficient, more secure and cheaper distributed security system. We present the essential concepts of a multiagent system, the role an agent plays in a distributed system, as well as the way a software agent communicates with a mobile robot. The paper also presents a practical real life example application that uses the Robotux system as an integrated solution for both security and "housekeeping" in a given environment.

Keywords: security system, multiagent robot system, software agent, mobile robot, Robotux.

1. INTRODUCTION

A classical surveillance security system uses video cameras and sensors to monitor the activity in a certain area. While this has proven to be a good approach, there are a few parts of the system which are error-prone. One of them is the so called *human factor* (explained bellow).

Most researchers use the term *agent*, to describe a sophisticated piece of software that acts autonomously in a distributed environment provided for it, solving a number of more or less complex problems. An agent is able to communicate with other agents within its habitat or other habitats.

The communication capabilities of an agent lead to the concept of a *multiagent system*, which is a distributed network collection of such software agents, working together to solve certain problems that are beyond the individual capabilities of each agent. Therefore a multiagent system is a collection of habitats where agents reside.

Without the use of a sophisticated software platform, mobile robots have a limited decision capability. They can be programmed to perform certain tasks, but they have no built-in capacity to communicate with each other. Thus, a certain software system must be provided in order for them to interact in an environment. The use of a multiagent system solves the problem, and adds a certain degree of "intelligence" to the robot.

A security system based on software agents and mobile robots has certain advantages over a classical security system. It can be cheaper, because the system is less-dependable on human personnel and usually keeps the maintenance costs low,

requiring only higher initial costs. It can be more efficient, since a classical system maintains a compromise between the number of video cameras used and the number of *dead angles* (portions of an environment where the camera can not "see"), while mobile robots can cover up to 100% of the environment's surface. Also, the security system algorithms can be less predictable than a human security officer, since a human being's activity can be (to some degree) predictable, while a robot can be programmed to just act "randomly" based on a good algorithm. The human factor also loosens up the security system, because of fatigue, routine, stress or plain simple poor observation.

In our example application we implemented *security* agents in charge of the environment's security, as well as *cleaner* agents in charge of housekeeping the environment.

2. IMPLEMENTATION

We propose a new approach, Robotux, a distributed security system, based on software Java agents and mobile robots.

Robotux is built using a customized Linux distribution, and it can be run from a live CD as well as from a hard drive. The entire platform is less than 220M, thus it fits on an 8cm CD (mini CD).

The chosen programming language for the security system software was **Java** from Sun Microsystems, due to its strong network facilities, which makes it an ideal platform for developing distributed software applications running on heterogeneous systems.

After testing several multiagent platforms, we selected the **Agents Development Kit** (ADK) version 2.1 from Tryllian BV, The Netherlands. **ADK** is built upon the Java programming language, Standard Edition, and offers strong agent development and deployment facilities, being both scalable and flexible. The inter-agent communication in ADK complies with a subset of the FIPA ACL standard (see [3]).

As a robot communication platform, we selected the **Player/Stage** project, an open source GPL development platform (see [5], [6], [7]) for robot and sensor applications. Both the cleaner and the security agents are implemented on Pioneer 2 robots from ActivMedia Robotics.

In order to clarify how our security system works, we present a model of our multiagent system in *Figure 1*.

The *Directory Facilitator* (DF) is in charge of "broadcasting" the map of agent services (each agent has a number of services – things they can do) over the network, as specified in the **FIPA Abstract Architecture Specification** (see [4]), chapter **5.6** *Directory services.* Exactly one DF agent *must* be present at each location (habitat), and each DF is a shared information repository where agents may publish their services and query other services of interest.

The *Database Connector* (DBCon) implements the "bridge" between the multiagent system and the database, which can be used to store information about and for the system. We used the popular MySQL database server suite for our database system.

The *Cleaner* agent implements the "housekeeper", which uses a mobile robot for actually cleaning the environment. Its job is to clean the entire environment, using a complex algorithm based on finding the shortest path combined with certain random elements, as well as supervisor commands, which will provide a higher unforeseeable pattern in case of a security breakthrough, thus making the system even more secure. The system has several security levels, with different action scenarios for each of them.

In case of a possible security breach (when a foreign object is detected), the cleaner is responsible to inform both the security and supervisor agents.

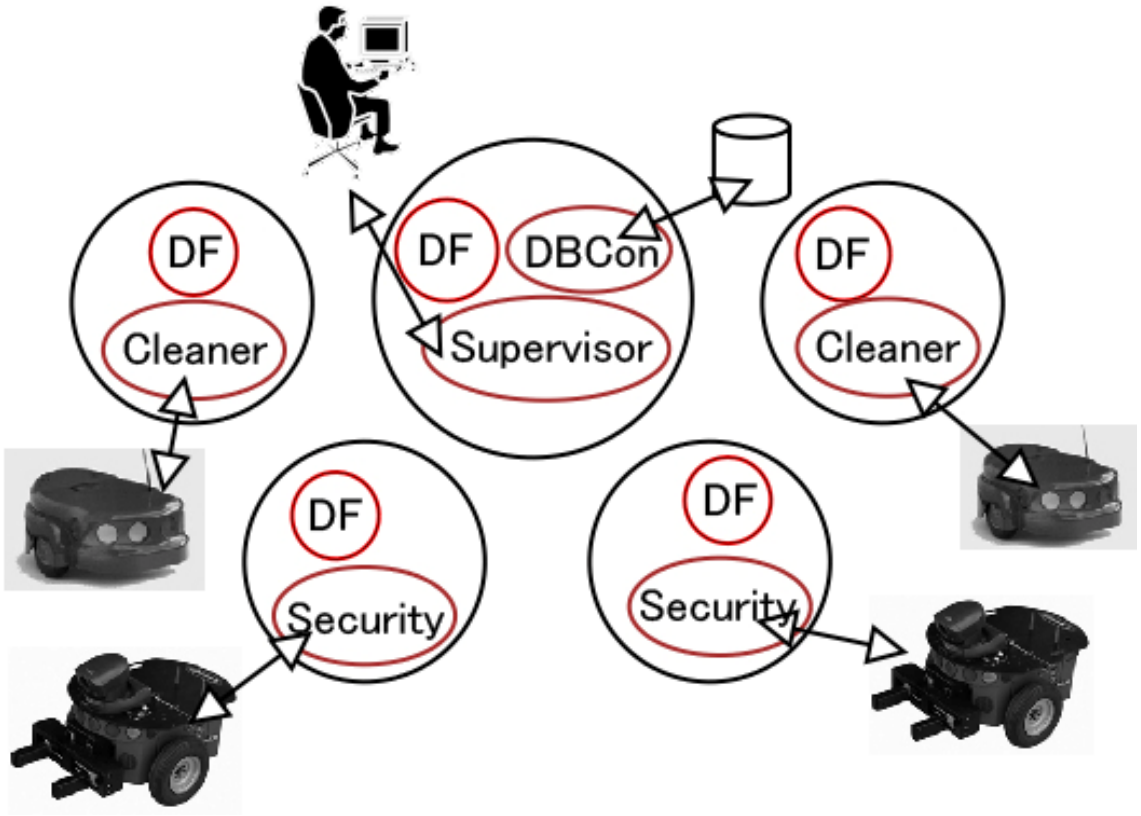


Figure 1. A model of the multiagent security system (robot pictures adapted from [8])

The *Security* agent is in charge of patrolling the environment with a certain movement pattern, and also to investigate possible security breaches. Like the cleaner agent, the security agent uses a mobile robot (more sophisticated in terms of sensors than the mobile robot used by the cleaner agent, as well as equipped with certain tools for intruder immobilization).

Finally, the *Supervisor* agent monitors the entire system activity and takes certain actions based on the behavior and information received from the cleaner and security agents. The supervisor agent's behavior and tasks may also be monitored by a human person.

The security and cleaner agents communicate directly with each other, but their behavior may also be influenced by the decisions made by the supervisor, which periodically receives information about the environment status.

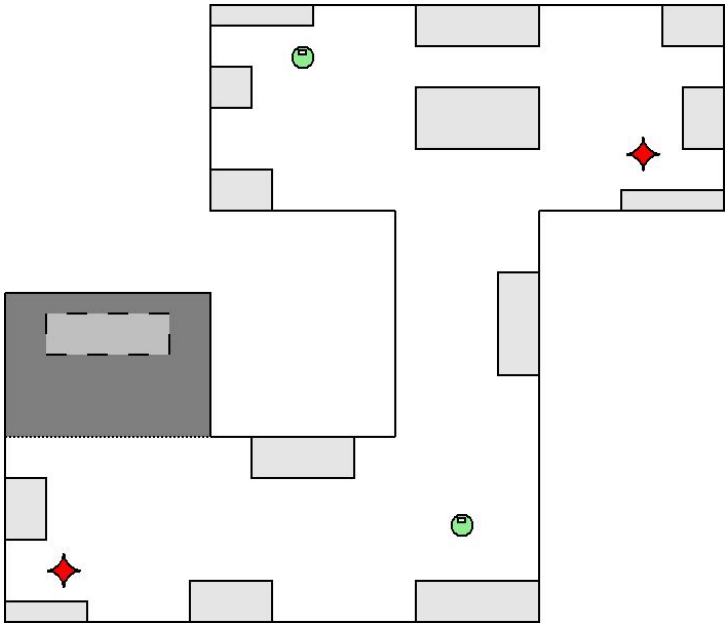
The multiagent system is fully scalable, new cleaner and security agents can be added dynamically, started in new habitats, as long as the new agents conform to the requirements stated above.

A certain degree of redundancy is available, meaning that an agent is able to take over the duties of any incapacitated agent in its group, if necessary.

3. EXPERIMENT

To demonstrate the effectiveness of our security system, we considered a real life application covering the interior of a museum. During the night, our system will have two primary tasks: cleaning the floors and securing the environment (exhibits) against burglars.

In *Figure 2*, we present a possible map of the museum for our application. The museum was recreated in one of our laboratories, using cardboard boxes for exhibits and a high-security area. To keep things simple, the environment has two rooms connected by a hall. One of the rooms contains a high-security area, shown in Figure 2 as the darker gray patch. The high-security area has simulated motion-sensors and laser grids activated on the floor controlled by the supervisor agent.



All the other light gray patches from Figure 2 are either glass cases or pedestals where museum exhibits are displayed to the public.

For simplicity, there are two *security* robots, represented in the figure by red stars, and two *cleaner* robots represented by green circles. For practical reasons, the number of cleaner robots is usually higher than the number of security robots.

Figure 2. A schematic map of the environment used in the project

The robots are connected to their appropriate agents and controlled by the agent software.

As we already mentioned, we used Pioneer-2 robots from ActivMedia Robotics for our project. The basic layout of a Pioneer-2 robot is presented in *Figure 3*.

The cleaner robots will move around the rooms, cleaning the floors of the museum, based on an internal map of the environment. The cleaner agent is also in charge of keeping the map up to date for its robot, receiving information from the other agents in the system.

The security robots must patrol the area to assure the exhibits' safety. They will also act to protect the cleaner robots upon necessity.

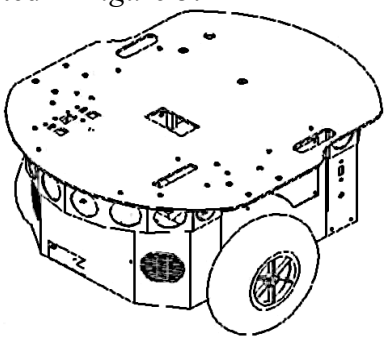


Figure 3. The basic layout of a Pioneer-2 robot (picture adapted from [8])

There are several counter measures a security agent can perform to immobilize a possible target, ranging from high frequency audio to powerful light flashes or even more. The activation of the stopping power on the security robots was also simulated and no actual devices were installed for safety reasons.

The sensors and grids in the high-security area can only be disabled by the supervisor agent who uses a complex algorithm based on random values as well as the position of the agents/robots in the system.

In order to be as cheap as possible, the cleaner robot was equipped with some basic sensors. The security robot on the other hand, has both longer range sensors and can also carry other intrusion detecting tools such as infrared heat detection sensors.

To better evaluate our system's reaction to a possible threat, we will take two case scenarios, one where the "threat" is just an object that has been moved from its regular position (or a new object on the scene) – **case scenario A**, and the other one where the unknown object is a human person – **case scenario B**.

A) The cleaner robot moves around in the environment using the map received from the cleaner agent. At some point, using his sensors, he detects an unknown object in the environment (an object is placed where it is not supposed to be). The cleaner agent has to evaluate the target, and see whether it is just a known object that has been moved with several inches, or a new object.

In case of a new object, the cleaner agent has the responsibility to inform both the supervisor and the security agents.

When a security agent receives notification about a possible threat, it will automatically send the security robot near the target site to investigate the problem. The cleaner agent can randomly perform one of the following two tasks: stay and monitor the target until the security robot arrives (the "play dead" tactic), or move around the object and continue its cleaning job (the "dissimulation" tactic).

The security agent reads the data from the security robot sensors and tries to accurately identify accurately if the target is "just" a new object, or a human person. The heat infrared sensor detection is a good method for scanning the target.

In case the target is a *safe* object (there remains room for the discussion of what is or is not "safe" in a certain situation), the security agent will inform all the other agents to update their maps and include the new object there.

B) The same tasks are performed by the cleaner and security agents, but in this case, the security agent evaluated the unknown object and decides that it could be a human person.

If the target is detected as a potential intruder, the security agent will alert the other security agents in the area (or possibly all of them) to send their security robots at the scene. Together they can perform a better scan of the target.

The speed and accuracy of the evaluation process depends on the type of sensors used on the robots.

After the evaluation process finishes, a quick decision must be made, the subject must be immobilized and the supervisor agent informed. It is the supervisor's task to inform the authorities of a possible burglar. No stopping power shall be activated before receiving approval by a human supervisor.

The high-security areas in the museum must also be cleaned. The supervisor decides when that action will occur, and will inform any available cleaner agent to move and clean the area. Any nearby security agent will also be informed, and a security robot will escort the cleaner robot at all times while it is cleaning a high security area.

Upon detection of an intrusion within a high-security area, all non-engaged robots will immediately move towards that area and immediately evaluate the situation and take the appropriate measures.

4. CONCLUSIONS AND FUTURE WORK

While we do not expect the Robotux project to become a new standard for a security system, there are several elements in it that we think a classical system could implement. A solution based on the use of software agents combined with mobile robots could not only be cheaper, since the need for human personnel diminishes, but it could be necessary in situations where a human's presence is undesirable (for instance in habitats where the temperature must be kept under control at all time, and a human person's body could affect in a negative way the environment).

The Robotux platform project has started from the need of having a platform for developing Java agent software for mobile robots, so it is and will be a continuous project for us.

Some things could be improved by the use of better, faster robots, or more powerful, accurate sensors. The robots run on hot-swappable batteries, but a system for detection of low battery status and automatic recharging using a regular power plug will be built.

Other areas in which further work needs to be done include improving the processing algorithms for sensors data, implementing a more powerful behaviour for the security agents as well as enhanced algorithms for robots formation grouping.

5. REFERENCES

[1] Ronald C. Arkin, (1998), *Behavior-Based Robotics*, The Massachusetts Institute of Technology Press 1998

[2] M. Isabel Ribeiro, Alessandro Saffiotti, (2002), *Lecture notes from European Summer School on Cooperative Robotics*, Institute for Systems and Robotics 2002

[3] *** *The Developer's Guide,* ADK 2.1 Edition, (2002), Tryllian Holding BV

[4] *** *FIPA SC00001, Abstract Architecture Specification,* http://www.fipa.org/specs/fipa00001/SC00001L.html

[5] Brian P. Gerkey, Richard T. Vaughan, Andrew Howard, (2003), *Player version 1.42rc2 User manual*, http://playerstage.sourceforge.net/doc/Player-manual-1.4.pdf

[6] Richard T. Vaughan, Andrew Howard, Brian P. Gerkey, (2003), *Stage version 1.3.3 User manual*, http://playerstage.sourceforge.net/doc/Stage-manual-1.3.3.pdf

[7] Andrew Howard, Nathan Koenig, (2004), *Gazebo version 0.3.0 User manual*, http://playerstage.sourceforge.net/doc/Gazebo-manual-0.3.0.pdf

[8] *** *Pioneer2/PeopleBot Operations Manual*, (2001), ActivMedia Robotics, LLC.