Annals of University of Craiova, Math. Comp. Sci. Ser.
Volume 36(2), 2009, Pages 109–117
ISSN: 1223-6934

A Mobile Agents Architecture for Robots Control

Claudiu-Ionuţ Popîrlan

Abstract. Constructing architecture and generating optimal control for robots are some of the heavily studied subjects in mobile agents applications. Currently, hybrid architectures offer the most widespread solutions for controlling intelligent robots. The aim of this paper is to design and simulate a modular and portable architecture that allows the development of robots control systems. A multi-level architecture based on the master/slave paradigm is presented. Its main components are mobile agents that interact through a communications framework. The results and simulations made on MobileSim simulator program are conducted to show the effectiveness of the proposed architecture.

2000 Mathematics Subject Classification. Primary 68T99; Secondary 65D17.
Key words and phrases. Mobile Agents, Master/Slave Architecture, Robots Control, Simulation.

1. Introduction

Agent-based approaches receive considerable attention in the literature. In daily life robots are used in many missions like space applications or planetary exploration. In these kinds of applications the robots control is very important.

Over the past few years, a new concept has entered the field of computer science. This is the concept of an agent. Some authors have used this concept to unify the field of artificial intelligence [1], while others see it as a new programming concept to extend the object concept [3]. The concept of an agent is strongly associated with localization. An agent often has a partial representation of the overall problem (which is synonym for the agents environment) that outlines the working conditions of the agent. Mobile agent is now a popular abstraction mechanism to construct adaptable distributed systems in convenient way. To develop a system using mobile agents, one has to overcome several new complexities (as well as security/ safety problems) in his/her code arise from the mixed use of mobility primitives. Over the last few years, numerous mobile agent systems [16, 14, 15, 12, 13] have been developed because this technology is promising for building customizable, adaptable and robust distributed systems.

Mackenzie describes in [4] an agent-based method for designing controllers for mobile robots. This method is based on the reactive robotics paradigm: the controller of the robot consists of a set of concurrently operating agents. Mackenzie defines an agent as a distinct entity capable of exhibiting a behavioral response to stimulus. Furthermore, in order to create hierarchical controllers, he defines an assemblage agent. An assemblage agent is a society of coordinated agents and can be part of some other assemblage agent.

Van Breemen presents in [5] an agent-based design method for solving complex control problems. He describes a method for modeling the structure of control problems in terms of partial problems and their interdependencies. His work provides a common language and a structured way to describe multicontroller systems. The method has an open character, such that parts can be added, removed or modified without disordering the operation of the remaining parts of the controller.

In this paper a mobile agents-based approach is proposed to design and simulate a control architecture for autonomous robots. It is accepted that the design of a control system for an autonomous robot is a problem with many facets. The robot has to carry out several tasks at the same time, such as achieving several goals simultaneously, actuate in a dynamic and uncertain environment, etc.

The approach consists of two agents types: the slave agent is responsible for the local problem (on the robots), and the master one is responsible for low-level control of the autonomous robots. These agents communicate via a communication module which is based on open agent architecture and is describe in Section 3.1.

2. Modeling Complex Control Problems

Control engineers are sometimes confronted with situations in which they have to design and implement real time control systems that are composed of a set of controllers instead of a single control algorithm. These situations occur for example when the control problem to solve is complex of nature, that is, when the problem can be thought of being composed of an interconnection of a set of simpler subproblems.

The solution of a practical control problem generally is a multi-controller: a controller that consists of a set of subcontrollers that are combined into an overall controller, such that when this controller is executed the overall performance specification is met. Depending on the way a control problem is solved, two different classes of problems can be defined.

Definition 2.1. ***Elementary control problem***
An elementary control problem is a control problem that is solved as a whole.

Definition 2.2. ***Compound control problem***
A compound control problem is a control problem that is solved by dividing it into a set of two or more partial problems.

The partial control problems that constitute a compound control problem are inherently coupled because they share the same parent problem. The type of coupling between the partial control problems determines the particular way in which the control algorithms have to be executed in order to have a well-behaved overall solution. Therefore, understanding how partial problems are coupled helps in making a decision about the way to decompose a problem and how to deal with the coupling effects.

Definition 2.3. ***Coupling relationship***
A coupling relationship is a property that holds between two partial control problems and which determines the restrictions with regard to executing the solutions, i.e. control algorithms, of the partial control problems.

The coupling relationships need only to be considered between partial control problems that belong to the same parent control problem. The fact that a compound control problem may consist of elementary control problems forms the basis for a recursive definition of control problems: *A control problem is an elementary control problem, or a compound control problem consisting of several partial control problems*.

Definition 2.4. ***Complex control problems***
A complex control problem is a hierarchical organized structure of elementary and

compound control problems. The latter consists of a set of partial control problems and a set of coupling relationships that exists between these partial control problems.

The class diagram shown in Fig. 2 illustrates this definition. As explained before, coupling relationships exist between partial problems.

2.1. Structuring the control problem. By taking into account the definition of a complex control problem, structuring a control problem consists of the following aspects:

- Decomposing the overall problem. The overall control problem is decomposed into a hierarchical structure of elementary and compound control problems. For each (elementary or compound) control problem a clear specification of the control objetive(s) in terms of the to-be-controlled variables is given. Such a specifications indicates, for example, how the controller should respond to disturbances and commands.
- Defining well-defined control problems. For each elementary control problem a model of the plant should be derived that is used, together with the objective specification, to design a control algorithm. Because the to-be-controlled variables cannot always be measured or directly controlled, candidate measured and controlled variables should be chosen that have a close relationship to the to-be-controlled variables.
- Specifying coupling relationships. Coupling relationships between the partial control problems are specified in order to take them into account when integrating the individual controllers.

3. The Proposed Mobile Agents-based Architecture

The architecture includes one master agent and several slave agents. The slave agent is responsible for the local problem (on the robots), and the master one is responsible for low-level control of the autonomous robots. Mathematical support for this master/slave architecture in order process inheritance knowledge bases was studied in [6] and [7]. Slave agents and mobile agent communicate with each other through the Communication Framework as shown in Fig. 1.

The mobile agents-based architecture consists of:

- 1 Master Agent(MA),
- n (n>0) number of Slave Agents(SA),
- distributed knowledge bases(KB1, KB2, ..., KBn)

To implement an actual agent-based framework, three additional parts are needed, namely: sensor components, actuator components, and a component that contains all parts of an overall multi-agent controller. This makes that this framework contains six functionally different components to construct multi-agent controllers:

- Sensor component: converts a value read from physical sensors into a meaningful number;
- Actuator component: converts a meaningful number into a value that is written to an physical actuator;
- Coordination object component: a component that coordinates a set of controller agent components;
- Elementary controller-agent component: a component that implements a locally operating controller;

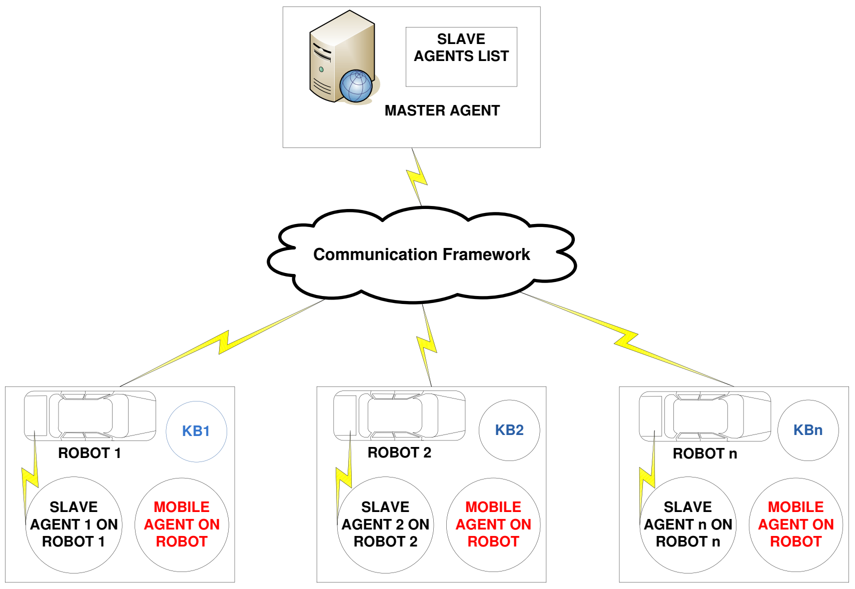


FIGURE 1. The mobile agents-based architecture

- Controller-agency component: a composite component that consists of a coordination object and a group of controller-agent components;
- Multi-agent controller component: a composite component that implements the overall controller and consists of sensor, actuator a coordination object and controller-agent components.

Figure 2 shows a class diagram of the relations between these different components. Both the multi-agent controller and controller-agency components are composite components, as they consist of other components. The remaining components are elementary; their functional behavior is implemented by specifying particular attributes and methods of the component. The interfaces of an elementary controller-agent and a controller-agency component are the same; both components can be considered as being general controller-agent components.

The interface of an elementary controller agent is made up of its inputs and outputs, and its activation request and acknowledge signals. The inputs and outputs are used to receive and send data from and to sensors, actuators and other controller-agents of the overall controller. The activation request and acknowledge signals are used to coordinate the behavior of the controller-agent with other controller-agents of the overall controller.

- activation request method to calculate the activation request signal of the controlleragent component;
- acknowledge method to receive ack signal from a coordination object component;
- initialize method of the local control algorithm;
- calculate method of the local control algorithm;
- update method of the local control algorithm.

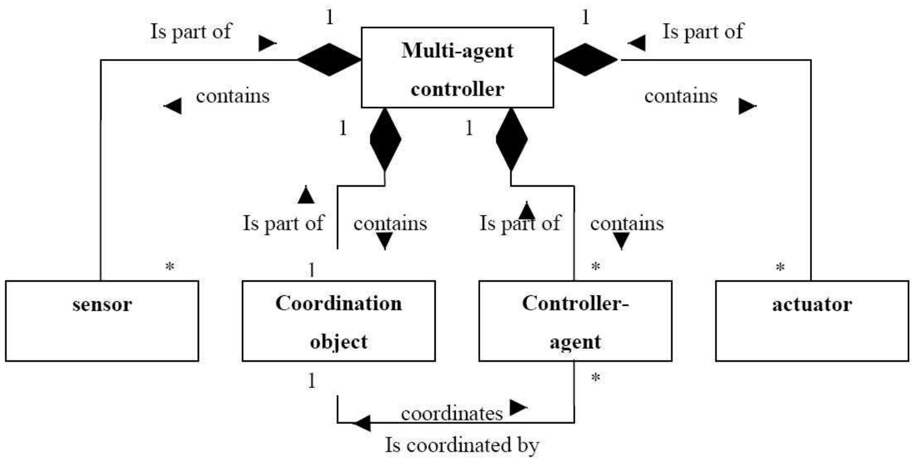


FIGURE 2. Class diagram of the six components of the architecture

3.1. The Communication Framework. The architecture must be able to make possible the agents communication, including the communication protocols. The communication protocols should allow message understanding and wireless transmission. Message understanding implies decision acceptance or rejection based on the collected sensors information and other mobile agents information. Mobile agents communication is based on sending (action) and receiving (perception) messages.

In proposed architecture, the communication framework has two levels:

- Level 1: communication between agents and sensors;
- Level 2: communication between agents.

Communication between agents and sensors (level 1) is in one sense: from sensor to agent. The agent receive informations from sensor and make decision.

Communication between agents (MA and SA), level 2, is done via messages. An agent can communicate with other agent by message passing. An agents that wants to communicate with another agent first has to create a message object, then send it to the target agent. A message object has a kind and an optional argument object. The receiver agent determine what to do by checking the kind of received message and get parameters as the argument object. For system implementation one can use a subset of a standard indicators ([11]) of the Knowledge Query and Manipulation Language (KQML):

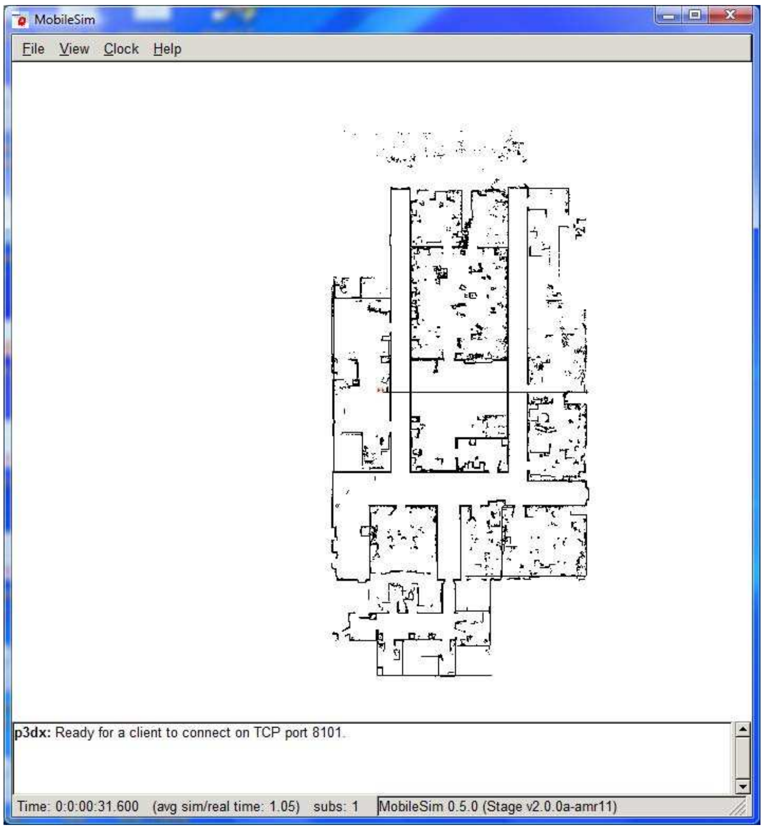
$tell$
$: content < expression >$
$: language < word >$
$: ontology < word >$
$: in-reply-to < expression >$
$: force < word >$
$: sender < word >$
$: receiver < word >$

This indicators can be used when an mobile agent found an object and announce its finding. The message agents of our system act somewhat like the active packets in the capsule tradition of active networking research; they are minimal programs that carry a payload of communications data across the network. They make their decisions about where they need to go based on information that routing agents, with which they share the network, accumulate and cache.

4. Simulation: Obstacle Avoidance by Robots

We have described so far our mobile agent-based architecture for autonomous robot. The presented design method aimed at a structured solution for a given task. The overall design procedure can be summarized as follows:

- Structure The overall control problem (robot control in our case) to solve should be decomposed into supposedly independent subproblems.
- Solve individual subproblems Each subproblem has to be solved by designing a controller-agent.
- Organize individual solutions The controller-agents are combined into one overall solution. Therefore is necesary to group controller-agents into controller-agencies and specify coordination mechanisms for each group of controller-agents. The overall solution is a hierarchical organized multi-agent controller.
- Implement the overall solution.



p3dx: Ready for a client to connect on TCP port 8101.

Time: 0:0:00:31.600 (avg sim/real time: 1.05) subs: 1 MobileSim 0.5.0 (Stage v2.0.0a-amr11)

FIGURE 3. The Virtual Reality Defined Environment

In this section it will be illustrated how our mobile agent-based architecture can be applied to control autonomous robots using MobileSim program ([18]). The aim of the control task was to make the robot wander around without hitting obstacles. The decomposition of the control problem derived into a set of simple behaviors, each of them solved by designing a controlleragent. These controller-agents were finally combined into one overall solution called multi-agent controller.

The explained approach is tested in MobileSim simulation environment. MobileSim is a test-bed for Pioneer robots and the programs are written with Aria framework. The programs that work correctly in simulator also work in the real life. Working environment is drawn by Active Media Mapper program and imported to MobileSim program. The model of the working environment is shown in Fig. 3.

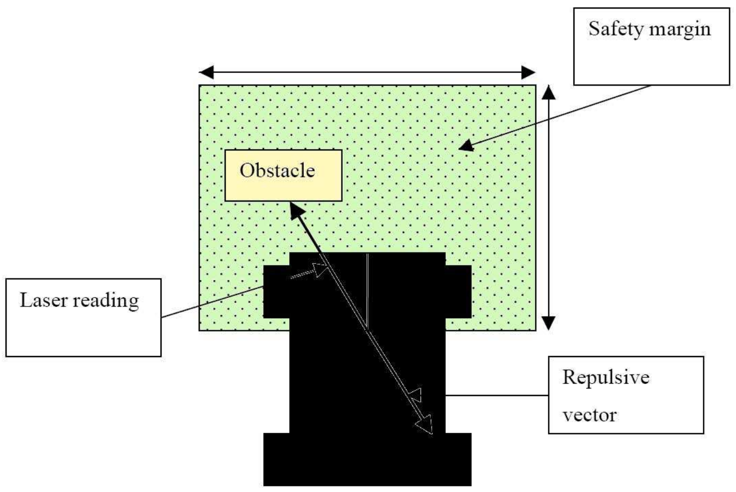
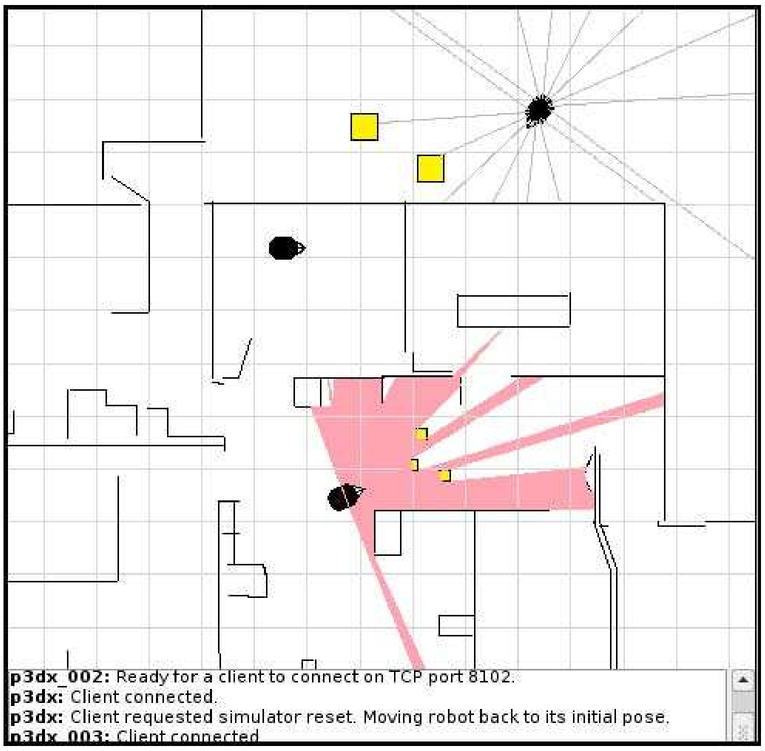


FIGURE 4. Autonomous Robot Avoid Obstacles

As was explained above, the overall control problem should be decomposed into a set of simple and independent subproblems, but there are no formal techniques to do that. These are the following:

- Straight forward: go straight. The Straightforward controller-agent always wants to get active.
- Wall-following: maintain the robot at a certain position in parallel respect to the wall. The WallFollowing controller-agent wants to get active when it detects a wall on the left side of the robot.
- Avoid obstacles: avoid any obvious obstacles. The AvoidObstacles controller-agent wants to get active when it detects an obstacle within the safety margin.
- Pass the door: go through the door. The Door controller-agent wants to get active when it detects a door on the left side of the robot.
- Stop: stop the robot. The Stop controller-agent wants to get active when the robot is about to collide with an object.

4.1. Solve individual subproblems. The individual subproblems have been solved by designing several controlleragents. We will give now a brief description of these controller-agents in terms of their inputs, outputs and particular algorithms used to accomplish their particular subproblems.



```
p3dx_002: Ready for a client to connect on TCP port 8102.
p3dx: Client connected.
p3dx: Client requested simulator reset. Moving robot back to its initial pose.
p3dx_003: Client connected
```

FIGURE 5. Autonomous Robot Avoid Obstacles

Algorithm 1 Avoid Obstacles

Input: Avoid Obstacles behavior takes its sensory information from Carthesian-Scanner. CarthesianScanner is an input object that reads values from Laser Scanner and converts those values from polar coordinates into carthesian coordinates.

Output: Wheelsystem: (velocity, angle). The output is expressed in terms of velocity of the robot and angle to turn.

Method: The internal representation used is that every laser reading represents a repulsive force. Is defined a safety margin as shown in Fig. 4, thus creating a vector among the direction of repulsion only if it is within the safety margin. The vector sum of the forces, suitably thresholded, determines in which direction the robot should move. Additionally, the velocity of the robot (0...1) is decreased proportionally, depending on how close the obstacles are from the robot.

At the first stage, the mobile agent gives the current grid position and the goal grid position to the path planning agent. Using this information and the map of the environment, the proposed mobile agents-based architecture constructs a path between the starting point and the goal point. And the constructed path is passed through communication channel to autonomous robots. These results can be visualized in Fig. 5.

5. Conclusions and future work

In this paper, a mobile agents-based approach was proposed to design and simulate a control architecture for autonomous robots. This architecture consists of two agents types: the slave agent is responsible for the local problem (on the robots), and the master one is responsible for low-level control of the autonomous robots. Anyway, the present study need to be extend for a dynamically changed environments. A increase on the grid size resolution so that the robots can navigate in dynamic environments, learning capabilities, etc. is also desirable.

References

[1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, 1995.

[2] J. Baumann, *Mobile Agents: Control Algorithms*, Lecture Notes in Computer Science, Springer, 2000.

[3] H. Parunak, Go to the ant: Engineering principles from natural multi-agent systems, *Artificial Intelligence and Management Science*, (1996).

[4] D. Mackenzie, A design methodology for the configuration of behavior-based mobile robotics, *Ph.D. dissertation*, University of Georgia Institute of Technology, (1996).

[5] A. Van Breemen, Agent-based Multi-Controller Systems A design framework for complex control problems-, *Ph.D. dissertation* , University of Twente, (2001).

[6] C.Popirlan and N.Ţăndăreanu, An Extension of Inheritance Knowledge Bases and Computational Properties of their Answer Functions, *Annals of the University of Craiova, Mathematics and Computer Science Series*, Vol.35, 149-170, (2008).

[7] C.Popirlan and N.Ţăndăreanu, Factorization of an Inheritance Knowledge Base, *Knowledge and Information Systems*, ISSN 0219-1377, submitted, (2008).

[8] G. Mester, Motion Control of Wheeled Mobile Robots, *4th Serbian-Hungarian Joint Symposium on Intelligent Systems, SISY*, 119-130, (2006).

[9] R. Gayle, A. Sud, M. C. Lin, D. Manocha, Reactive Deformation Roadmaps: Motion Planning of Multiple Robots in Dynamic Environments, *Intelligent Robots and Systems, IROS 2007. IEEE/RSJ International Conference*, 3777-3783, (2007).

[10] T.-J. Pan, R.C. Luo, Motion planning for mobile robots in a dynamic environment with moving obstacles, *Robotics and Automation, IEEE International Conference Proceedings*, Volume 1, 578 - 583, (1990).

[11] T. Finin, Y. Labrou, J. Mayfield, *Software Agents: KQML as an agent communication language, invited chapter in Jeff Bradshaw (Ed.)*, MIT Press, Cambridge, 1997.

[12] D. Kotz, R.S. Gray, Mobile Agents and the Future of the Internet, *ACM Operating Systems Review*, Volume 33(3), 7-13, 1999.

[13] Q. Wenyu, S. Hong, X. Defago, A Survey of Mobile Agent-Based Fault-Tolerant Technology, *Parallel and Distributed Computing, Applications and Technologies*, Volume 5, 446-450, 2005.

[14] H. Peine, An Introduction to Mobile Agent Programming And The Ara System, *Technical report ZRI-Report 1/97, Department of Computer Science, University of Kaiserslautern*, Germany, 1997.

[15] R.S. Gray, D. Kotz, G. Cybenko, D. Rus,D'Agents: Security in a multiple-language, *In Giovanni Vigna (Ed.) Mobile Agents and Security, Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 1998.

[16] D. Wong, N. Paciorek, T. Walsh, J. DiCelie, M. Young, B. Peet, Concordia: An infrastructure for collaborating mobile agents, *Lecture Notes in Computer Science*, Volume 1219, 86-97, 1997.

[17] D. Johansen, R. van Renesse, F.B. Schneider, Operating System Support for Mobile Agents, *Technical report, Department of Computer Science, Cornell University*, USA, 1994

[18] `www.activrobots.com/SOFTWARE/aria.html`

(Claudiu-Ionuţ Popîrlan) University of Craiova, Faculty of Mathematics and Computer Science, Department of Computer Science, 13 Alexandru Ioan Cuza Street, Craiova, 200585, Romania
E-mail address: `popirlan@inf.ucv.ro`