Multi-Robot Management Framework based on the Agent Dual-Space Control Paradigm

Stanisław Ambroszkiewicz,
Institute of Computer Science,
Polish Academy of Sciences, Warsaw
sambrosz@ipipan.waw.pl

Krzyszof Cetnarowicz and Wojciech Turek
Institute of Computer Science
AGH University of Science and Technology, Krakow, Poland
cetnar@agh.edu.pl, wojciech.turek@agh.edu.pl

Abstract

Some new ideas concerning agent and multiagent systems in the context of robotics are presented. These ideas are based on long term research and experiences in agent theory (M-agent architecture) and technology, see (Cetnarowicz 1996), (Ambroszkiewicz & Cetnarowicz 2006). The M-agent (Cetnarowicz 1996) architecture may be used for the taxonomy of multiagent systems depending on the nature of the environment. If the environment is a cyber-space, MAS contains software (possibly mobile) agents. If the environment is a real-space, there are devices (like mobile robots) that may communicate and interoperate. Following this idea, we may say that robot (considered as autonomous intelligent being) is in the cyber-space as a software application (M-agent), whereas the real robot is reduced to a device (tool) performing a specific service, and changing, in this way, the real-space. The presented approach leads us to the archtecture of the multi-robot system, and then to framework for multi-robot system development The framework assumes, that a robot is not a pro-active participant of the system. It is considered to be a tool, which must support several required functionality, and can support some additional features. The mentioned approach to the multi-robot system development and the application of the framework are presented in the paper.

Agents and robots

Research into cooperation of multiple mobile robots received significant attention over last years (Ota 2006). The most popular approach adopts Agent-Oriented design paradigm as a control software development methodology. Multi Agents Systems (MASs) offer several advantages, which are very desirable in complex control systems, like maintainability, extensibility, parallelism and fault tolerance.

Certain similarities between software agents and autonomous mobile robots, like autonomy, pro-activity, communication or mobility, can be observed very easily, although meaning of some of these features is slightly different. Therefore numerous works, concerning different applications of mobile robots (Hsu & Liu 2005; Candea *et al.* 2001; Friedrich, Rogalla, & Dillmann 1998), define software agent as a control program responsible for managing a single robot. Several such agents create a MAS, that is used for solving a task.

Although this approach seems very natural, it creates significant drawbacks:

- knowledge about a problem must be distributed among all agents,
- complex protocols must be developed to support coordination between agents, while performing tasks, which require cooperation,
- if an agent is to work directly on a robot, an on-board computational units and communication devices must be very efficient.

In this paper authors present a different approach to design of Multi-Agent Systems for control of groups of mobile robots.

The M-agent architecture (Cetnarowicz 1996) may be used for the taxonomy of multiagent systems depending on the nature of the environment. If the environment is a cyber-space, MAS contains software (possibly mobile) agents. If the environment is a real-space, there are devices (like mobile robots) that may communicate and interoperate. Designing MAS for such kind of environment is not easy - direct mapping of a robot into a single agent is not a good solution ((Ambroszkiewicz *et al.* 2000), (Ambroszkiewicz & Cetnarowicz 2006)).

Depending of the complexity of the models of the environment we may distinguish the following cases. If the environment is simple and fixed, then also agent can be simple and built according to the reactive architecture and a single agent may be associated to a single robot. However, the environment is usually rather complex and the agent must be cognitive so that its mind should according to the deliberative architecture. In this case, it is difficult or even impossible to store the (software) agent on the robot's onboard computer. The agent may be situated in the cyber-space (as a process running on a remote host), controlling the robot from the host via wireless communication devices (Fig. 1 a)).

Following this idea, we may say that robot (considered as autonomous intelligent being) is in the cyber-space as a software application (built as a M-agent), whereas the real robot is reduced to a device performing a specific service, and changing, in this way, the real-space ((Turek, Marcjan,

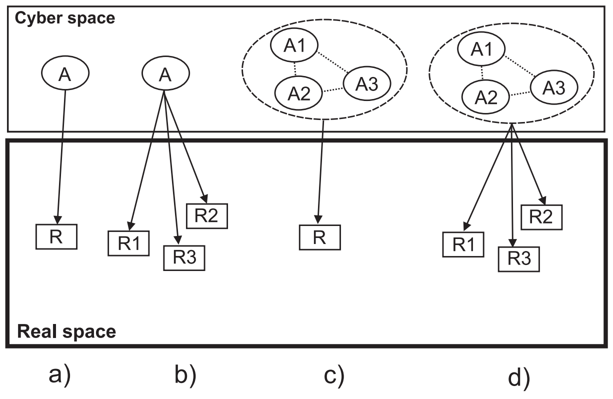

Figure 1: Situating agents and robots in real-space or in the cyber-space.

& Cetnarowicz 2006). Moreover, a multiagent system in the cyberspace may use robots (as services) situated in the real-space (Fig. 1 b, c)). Finally, multiprofile M-agent situated in the cyber-space may control a multi-robot system, see Fig. 1 d). In this case each profile is responsible for controlling separate service performed by a device (robot) in the real-space. The agent in the cyber-space we call agent-robot and the robot in the real-space we call tool-robot or service-robot.

Presented paradigm is a basis for an Agent-Based Framework described in this paper. The framework is designed as a solution to an abstract Task Execution problem, where a group of robots must cooperate in order to fulfil tasks, which continuously appear in their environment. Environment is considered known,

Robot as a tool

The framework assumes, that a robot is not a pro-active participant of of the system. It is considered to be a tool, which must support several required functionalities, and can support some additional features. A robot must obey any order or function call, that is sent to it; its reactions should be deterministic and predictable on visible state. Features that must be provided by each robot are (fig. 2):

- report abilities
  - static features (size, capacity, ...)
  - dynamics characteristics (velocities, accelerations, ...)
  - possessed tools - tasks execution capabilities
- report state
  - battery low, fuel low,....
  - failures detected, ...
- move according to precise orders (orders like)
  - linear velocity,
  - angular velocity,
  - curvature characteristics,
  - time
- move to a given destination with sensor-based collision avoidance
  - linear velocity,
  - ETA
- perform tasks, according to reported abilities
- report location
- if no external localization system present - calculate location and its accuracy (SLAM).

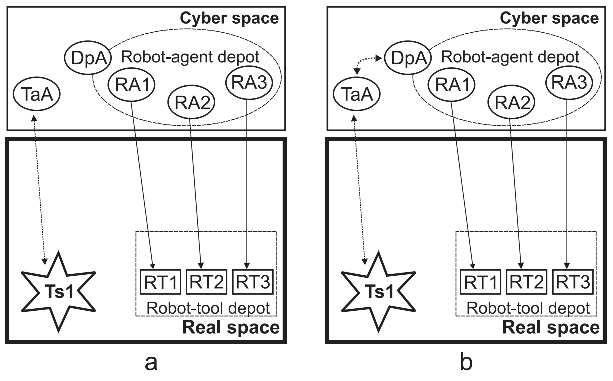

Figure 2: Example of the cooperation among agents and robots with depot service - negotiation to get in rent a robot-tool.

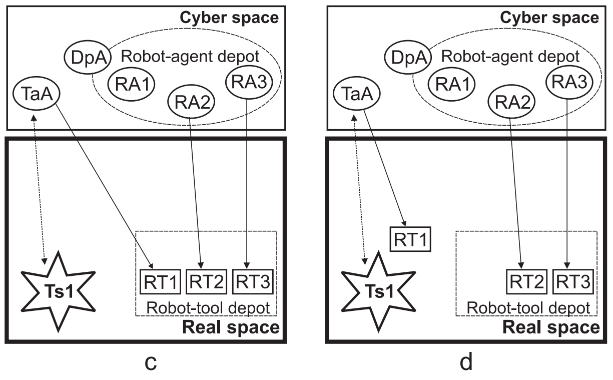

Figure 3: Example of the cooperation among agents and robots with depot service execution of a given task by the robot-tool.

The base principle of the agent dual-space paradigm: the agent in cyberspace is a robot (called agent-robot), robot in real space is only a tool or service (called robot-tool). We have following agents and robots

- TaA - Task-Agent that is to complete a given task,
- DpA - Depot-Agent is an agent managing depot of robots
- RA - Robot-Agent is an agent charged with control of a given robot,
- RT - Robot-Tool is a robot working in a real space

We can consider following cooperation among agents and robots:

- Robot TaA (Task Agent) find a task Ts1 to be completed (fig. 2 a).
- Robot TaA due to the negotiations with depot manager agent DpA get in rent a real robot - robot-tool (RT - Robot-Tool) to complete a given task (fig. 2 b).
- Depot manager agent DpA selects a robot-tool (for example: RT1) that is just ready to work, and transfers it (the control over it) to the task agent TaA1. Task agent TaA1 starts to control and manage the robot-tool RT1. (fig. 3 c).
- The task-agent TaA with the use of robot-tool RT1 complete the task T1 (fig. 3 d).

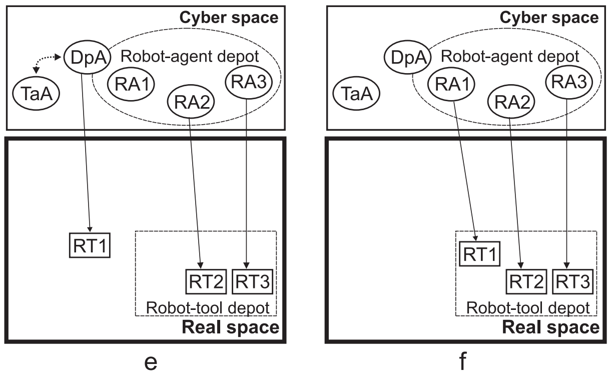

Figure 4: Example of the cooperation among agents and robots with depot service - returning of used robot-tool operation.

- When task T1 is completed task-agent TaA returns the controlled robot-tool RT1 to the depot-manager agent. (fig. 4 e) .
- Depot-manager agent DpA selects a free service-agent (for example RA1) and transfers to it control over the robot-tool RT1. Agent RA1 starts the maintenance procedures with the robot-tool RT1 (for example: filling batteries, cleaning etc.) (fig. 4 f).

Robots, that are not performing any task at a particular moment, are to be placed in a Robot Depot, which is a separated place in the environment. A depot-agents are capable to perform service operations on robots (robot-tools) stored in the depot.

A depot-manager agent (with a group of subordinate agents) is an active management element, that is responsible for the following operations performed in the depot:

- organization and matinee of the depot,
- displacement of robots in the depot
- service operations (with queuing for those) necessary for robots in depot
- collecting information about robots in the depot to realize negotiation with Task Agents (about real and required robot's features - appropriate robot for a given task),
- queuing of robots for tasks,
- serving information about Task Agents currently owning and using robots taken from the depot.

Task detection is a particular task for multi robot system.

A task-execution agent (TaA) is an active element of the system, that is responsible for the realization of a given task. Each task is characterized by:

- location
- priority - used for queuing for robots in depots
- type and number of tools required for fulfilling
- information enables parallelism of execution (a given task may be performed faster, if more robots is involved)

A Task Agent must to be able:

- find a robot (a group of robots), that will perform a given task,
- priority-based negotiations with depot-manager agents,
- asking other Task Agents for robots that are no longer necessary, or for estimated time when the reserved robot could be released,
- Navigate robots to a given location,
- Performed a given task with the use of robots.

The Task Agent desires to finish a given task as soon as possible, therefore it must gather optimal number of robot around the task in optimal time (using negotiations).

After a given task is finished, Task Agent must "get rid of robots" returning them to the depot (or depots):

- navigate robot to the depot (closest or appropriate to the type of the robot)
- transfer control over the returning robot to other Task Agents of the depot.

As an example of the robot management during the task execution we can consider a transfer of a given robot i real space by particularly difficult area (e. g. narrow gate) (fig. 5).

The transfer of the robot by the gate may be presented in the following scenario:

- Task agent TaA is coming up to the gate with its robot-tool RT1 and negotiates with area agent AA the passage (fig. 5 a).
- AA agent assigns a free robot-agent (RA1) that is capable to complete the transfer (fig. 5 b).
- RA1 agent drives (successfully) the robot-tool by the gate (fig. 5 c).
- AA agent put back the control over the robot-tool RT1 to the agent TaA that may continue the execution of the task (fig. 5 d).

Presented examples may give an idea how the presented approach to the agent - robot relation works in realization.

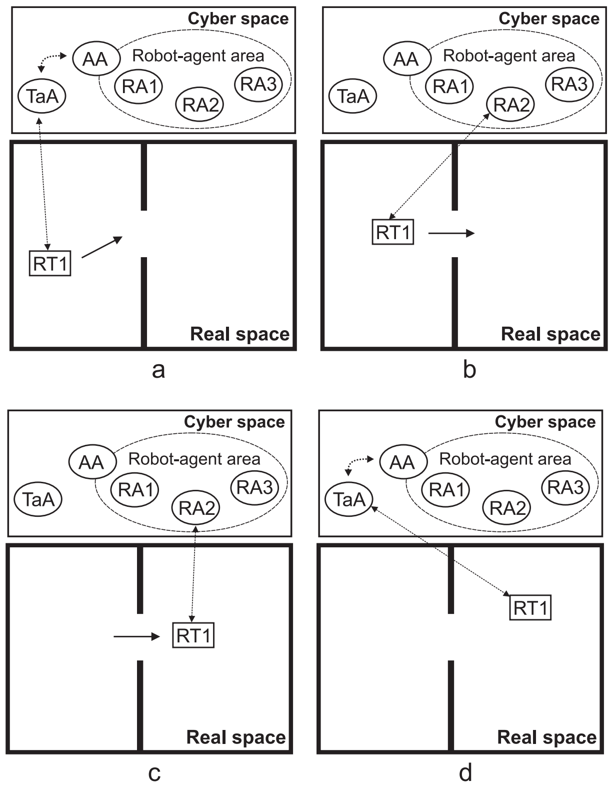

Figure 5: Example of the transfer of a robot by the particular area (gate)

Environment representation and an Area Control Agent

To realize presented operations a solution of navigation and motion coordination problem is needed. We can explain the problem and its solution using an example (fig. 6 a). The considered environment is divided into rooms. It is assumed, that each robot can safely and robustly traverse a single room using its sensor-based collision avoidance algorithm.

A typical reactive algorithm can deal with:

- convex obstacles
- other robots or different, small moving obstacles

More complex areas are controlled by area-control agents, (Area Agent AA) which are responsible for motion and displacement management for multiple robots in the assigned part of the environment.

An area agent (AA) should implement an advanced planning strategy, to ensure safe and robust motion.

Exemplary types of areas , that should be controlled by Area Agents:

- narrow passages
- junctions of corridors
- doors,
- access restriction passages
- elevators

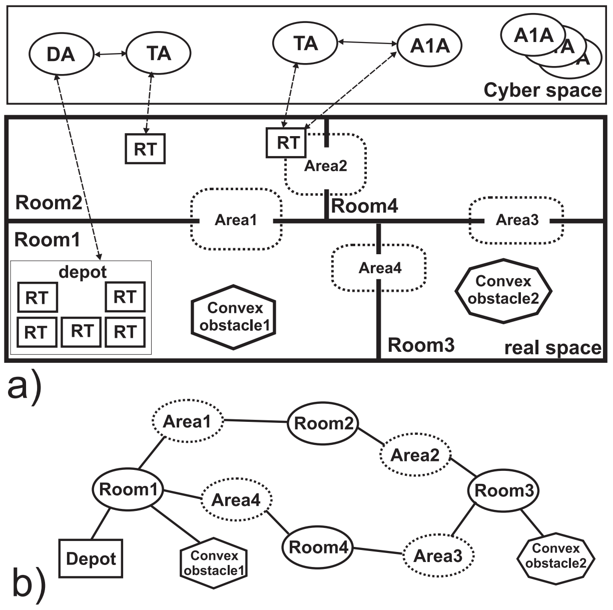

Figure 6: a) Exemplary environment with 4 rooms and 4 areas , that require management; b) a graph (map) for the environment

Area agent must be able to manage movement of robots, but also is to inform task agents (TaA) about properties of controlled areas. The task agent would require giving answers to questions like:

If robot R arrives at time t0 from room n, - when can it be transferred to room m?

Rooms and areas controlled by area agents AA may be represented by a relatively simple graph, which is an abstract model of the environment (fig. 6 b).

The graph (map) can be considered a global, common knowledge base about environment topology. It should be implemented as a service, that supports information about:

- location affiliation - (x,y) is located in Room n or area agent AA,
- possible paths between locations,
- each room edge can be associated with a constant value representing length.

The global path planning and transfer by the path may be realized by special services. Task agent must be able to find optimal paths for its robots. The path planning algorithm is based on the graph described before - all possible, or 'probably best' paths can be received from the map service. For each of the paths an approximated travel time must be calculated due to the information provided by area agents about estimated passing time.

The execution of planned path consists of autonomous and separately managed parts of the environment. In rooms a robot must be able to find in autonomous way its way to a given destination (point, or an area). However, in areas control over every robot is passed to the area agent AA.

The Multi-robot Management Framework

A special tool for testing presented approach has been developed (system called RoBOSS (Turek *et al.* 2005)). The system is composed of development tool for the multiagent system creation and multi-robot system simulator that is used to verify developed multiagent software. When the multiagent system is created and verified, the simulated robots are replaced by real robots in a real environment, and system may start to work (fig. 7).

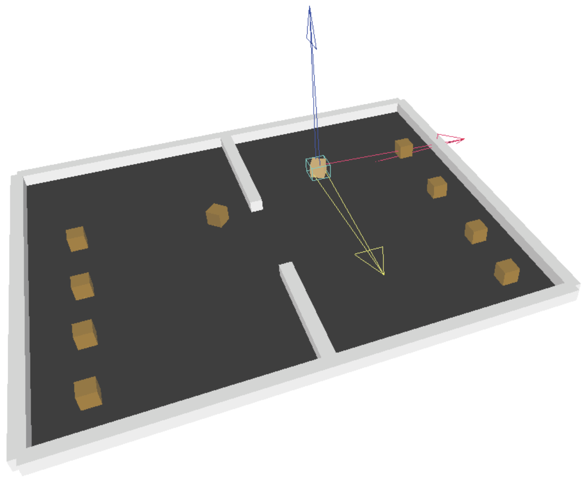

Figure 7: Example of the framework for multiagent-multirobot system development

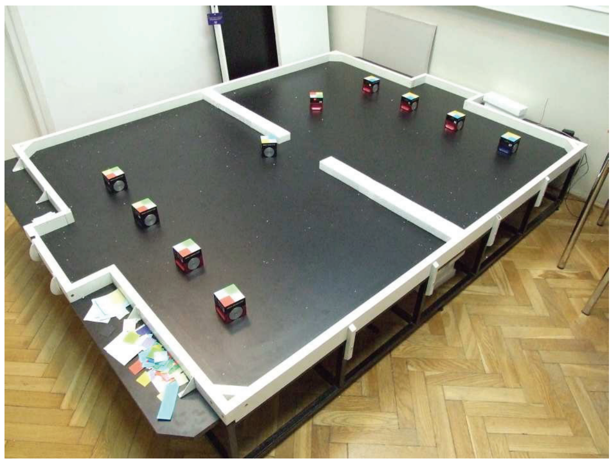

Figure 8: Example of the cooperation among robots programmed with developed framework

The following experiment has been implemented and tested in simulation and in reality, using hardware robots (fig. 8):

- basic task execution:
  - a task is detected, a Task Agent is created,
  - the Task Agent asks Depot Agents, find a robot, that will perform the task
  - the Task Agent plans a path to the task using the map service and communicating with an area agent, responsible for motion coordination in a narrow passage
  - the robot moves to required location, the task is successfully executed,
  - the Task Agent plans a path back to the depot;
- variants
  - multiple robots are assigned to realize a given task,
  - overtaking a robot, that is going back to a depot to use it in the realization of the next task.

These tests show, that the method is correct and the system can handle numerous tasks without unjustified delays or deadlocks. Unlike the classic approach, where each robot is individually controlled by an agent, implementation of motion and task execution coordination was relatively simple.

Conclusions

The approach presented in this paper is based on a concept of considering a hardware robot as a tool, rather than an proactive participant of a multirobot system. This logical separation of an agent controlling a robot from the robot itself seems to simplify design of an agent-based multirobot system.

The framework, implemented using this paradigm, has several features, that are very desirable in large scale control systems.

- Whole problem is divided into smaller sub-problems, non of which requires performance larger than can be offered by available devices:
  - hardware robots perform relatively simple and well defined tasks (reactive motion, effectors usage and self control), therefore on-board computers do not have to be very powerful,
  - advanced path planning and motion coordination is applied only in selected areas of limited size,
  - global path planning operates on a simple, very high level model of the environment.
- The solution is naturally distributable and scalable, which is common to most agent-based systems.
- Whole life cycle of the system is considered, including robots service needs and idle periods.
- High availability is achieved due to separation of control - failure of a robot can be easily handled, and does not cause task abandoning or significant delays.

It is difficult to perform tests of scalability using real hardware, therefore only simulations of large scale multi-robot systems can be analysed. Results of the approach seem promising, and further research into application of the framework to more specific task will be carried out.

Acknowledgments

This work has partly been supported by the grant MEiN Nr 3 T11C 038 29.

References

Ambroszkiewicz, S., and Cetnarowicz, K. 2006. On the concept of agent in multi-robot environment. *Innovative Concepts for Autonomic and Agent-Based Systems. Lecture Notes in Computer Science Workshop on Radical Agent Concept, Washington DC. USA. Springer-Verlag Berlin-Heidelberg* 3825:135–146.

Ambroszkiewicz, S.; Cetnarowicz, K.; Kozlak, J.; Nowak, T.; and Penczek, W. 2000. Modelling agent organizations. *Proceedings of the conf. Intelligent Information System. Advances in Soft Computing, Springer Verlag Berlin* 135–144.

Candea, C.; Hu, H.; Iocchi, L.; Nardi, D.; and Piaggio, M. 2001. Coordination in multi-agent robocup teams. *Robotics and Autonomous Systems* 36:67–86.

Cetnarowicz, K. 1996. M-agent architecture based method of development of multiagent systems. *Proc. of the 8th Joint EPS-APS International Conference on Physics Computing, ACC Cyfronet Krakow Poland*.

Friedrich, H.; Rogalla, O.; and Dillmann, R. 1998. Integrating skills into multi-agent systems. *Journal of Intelligent Manufacturing* 9:119–127.

Hsu, H. C.-H., and Liu, A. 2005. Multiagent-based multi-team formation control for mobile robots. *Journal of Intelligent and Robotic Systems* 42(4):337–360.

Ota, J. 2006. Multi-agent robot systems as distributed autonomous systems. *Advanced Engineering Informatics* 20(1):59–70.

Turek, W.; Czyrnek, D.; Marcjan, R.; and Cetnarowicz, K. 2005. Roboss - an universal tool for robots modelling and simulation. *CMS'05, Plenary lectures and special session papers : Computer Methods and Systems. Proceedings of the conference. AGH University of Science and Technology, Cracow, Jagiellonian University, Cracow University of Technology. Oprogramowanie Naukowo-Techniczne, Krakow* 1:347–354.

Turek, W.; Marcjan, R.; and Cetnarowicz, K. 2006. Agent-based mobile robots navigation framework. *ICCS 2006, Lecture Notes in Computer Science, Springer-Verlag, Berlin-Heidelberg* 3993.