How to Make a Good System from Imperfect Components

Andrej Lúčny
Department of Applied Informatics
Faculty of Mathematics, Physics and Informatics, Comenius University
Bratislava, Slovakia
lucny@fmph.uniba.sk

Abstract — **We present a control system of LEGO robot which is compounded from several different and competitive behaviors. It has been used for training of master course multi-agent systems to present to student a real example how a good overall behavior can emerge from imperfect individual behaviors.**

Keywords – robot, behavior, education

I. INTRODUCTION

Cheap and widely available robots are popular teaching aids. They are not only attractive for students, but they enable teacher to show many ideas from many domains in real operation – not only those ones tied with education of robotics. Of course, teacher has to invest his effort to such presentations and exercises and must have enough knowledge and/or effective support from field of robotics.

We present exactly such approach here: we use simple LEGO robot to demonstrate profit of agent-oriented programming. Agent-oriented programming is a method of software development derived from field of multi-agent systems. While multi-agent systems are necessarily distributed systems, agent-oriented programming applies the same modularity and organization for development of non-distributed systems. Good example of such system is a system for controlling a robot. This gives us opportunity to employ robotics in education of multi-agent systems.

The main profit of multi-agent systems (besides handling of distributed character of their applications) resides in ability to generate better overall behavior of system than the behaviors of its individual components can provide in total. (This ambition makes domain of multi-agent systems close to artificial intelligence). It is relatively difficult to demonstrate this idea hence range of usual applications typical for multi-agent systems overcomes conditions which teacher can emulate at classroom. Therefore we employed robot controlled by system with analogical modularity and organization. This enabled us to show the profit very clearly.

II. TECHNICAL ACCOMPLISHMENT

A. Hardware and software

We have used LEGO Robolab. We have build robot typical for line following equipped with two motors and tracks, one light sensor and one touch sensor; all controlled by RCX LEGO brick (Fig. 1). We have used Lejos firmware and development kit for RCX (http://lejos.sourceforge.net). We employed robot scene for 'Line Follower' contest, which is part of e.g. ISTROBOT contest.

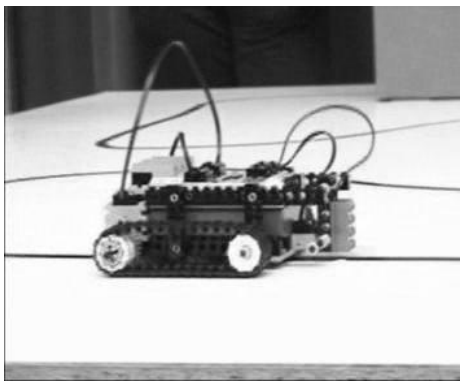

Fig. 1. Robot following line

B. Problem specification

Robot, architectural framework for building its control software and individual components for its development are given to students. Students are asked to fill the components gradually into the framework and put the robot into operation. Then the conditions in the scene are changed and students observe how robot is able to treat the modified situation. The scene difference is chosen to present that robot is operational also under changed situation but just due to multi-agent features of the employed framework.

Particularly, the framework is based on indirect communication among agents through a backboard ([3], www.agentspace.org). It enables any agent to interfere communication among other agents.

The task is similar to 'Path follower' category of popular contest ISTROBOT ([2], www.robotics.sk). Robot has to follow a line, but it has also to overcome a brick which is put on the line.

C. Problem solution

Usual solution of this problem is based on consecutive execution of two behaviors: on following the line and avoiding the brick.

This work was supported by VEGA grant VEGA 1/0280/08

Following is based on a single light sensor. We move only one track at same time. We move with a track while the light sensor detects line. Then we stop the track and start to move the other track. This simple behavior causes that the robot moves on line when the line is present and the robot stops otherwise.

Avoiding is based on blind sequence of proper movements started by touch of the touch sensor. The size and direction of the consecutive movements are selected due to dimensions and position of the brick. (It depends also on the current capacity of batteries little bit.)

The easiest way how to combine the two behaviors is a pure pipe-line (Fig. 2). This solution works only for one brick and it fails at all when we use a brick with other orientation (e.g. we turn the brick to perpendicular position).

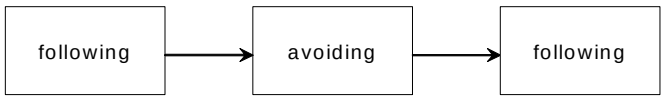

Fig. 2. Traditional combination of behaviors

A novel solution we like to present to students is based on parallel combination of the two behaviors (Fig. 3). Both Following and Avoiding are parallel modules with own control (with own threads).

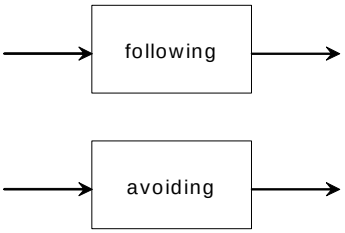

Fig. 3. Novel combination of behaviors

Following is active when line is present (and when we has just lost it), while Avoiding is active after the touch sensor is activated. Avoiding has a higher priority than Following at the moment of the touch. Its priority is lower otherwise (mainly we need to concern the moment when the line is found again). This cooperation is provided the framework.

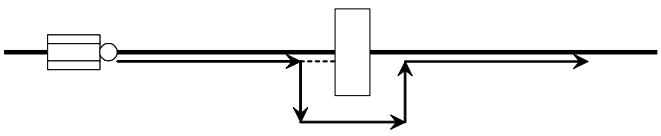

Fig. 4. Solution of the normal situation

This solution can handle the situation when the traditional approach works (Fig. 4). However, when we modify the brick orientation, it works too (Fig. 5). This fact is surprising and makes a strong impression to students showing them how our expectation can fail when we deal with systems which internal structure resemble to organization of living systems – even in such simplified case.

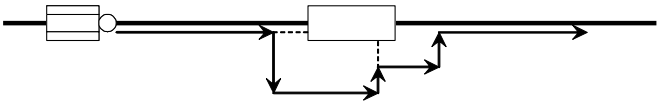

Fig. 5. Solution of the modified situation

Of course, we see no prodigy here. Just the behavior Avoiding designed for single use has been used two times due to another touch we did not expected. This is very clear when we see it, but it is hardly expectable until we see it.

In this way the non-traditional framework enabled us to create system which overall behavior overreaches the sum of capabilities of its individual components. We have built a good system from less perfect components.

III. EDUCATIONAL ASPECTS

This example is very simple, but just such examples are suitable to be shown during two hours of student exercise.

We have worked with group of 10 students (Fig. 6) divided to two groups. Lejos and windows shell extension for compiling java source files and running compiled classes (we avoided any IDE in this way) have been available for students to simplify use of equipment and to save enough time for dealing with application. At the first, students have to put to operation the traditional solution. Then they had to use the novel framework known to them also from previous exercises. Each group had a responsible person selected by teacher. One group succeeded to carry out the experiment, one failed – mainly due to lack of time. Even for this simple example time is critical and everything which has no direct relation to the main goal must be prepared in prior to exercise.

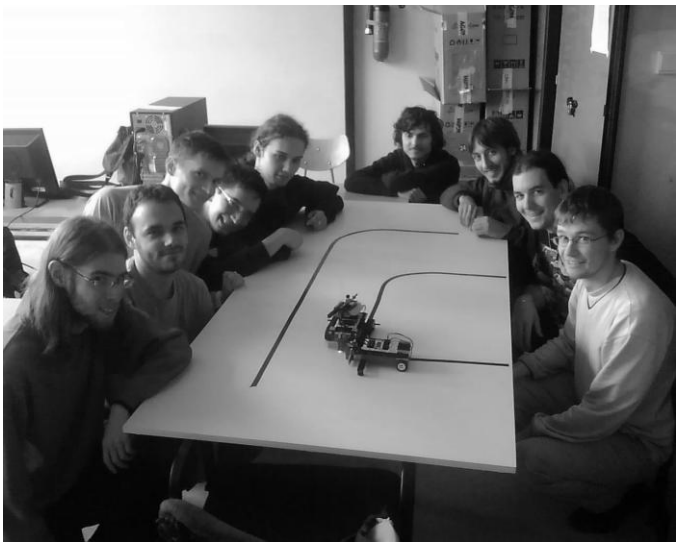

Fig. 6. Tested group of students

IV. CONCLUSION

We have presented one example how to employ robots in education. The aim of this employment was to show to students a profit from non-traditional organization of software. The example was tiny but suitable for student exercise. In this way we presented a principle from domain of multi-agent systems which could be hardly presented in a traditional way.

ACKNOWLEDGMENT

We thank to Jan Sefranek, associated professor at FMFI UK Bratislava for his long-term support of robotics in education at our Department of Applied Informatics.

This paper was created with support of VEGA 1/0280/08 “Application of multi-agent modularity for development of control systems and computer models”.

REFERENCES

[1] Bagnall, B.: Maximum LEGO NXT. Variant Press, Winipeg, 2007

[2] Balogh, R.: Popularization of the robotics, In DidInfo 2008. 14th International Conference. FPV UMB, Banská Bystrica, 2008

[3] Lúčny, A.: Building Complex Systems with Agent-Space Architecture. Computing and Informatics, Vol. 23 (2004), pp. 1001-1036