Disbotics
Himanshu Mongia, Manuel Esberger
Vienna Institute of Technology, manuel.baskhiron@student.tgm.ac.at, hmongia@student.tgm.ac.at

DISBOTICS

DISBOTICS – THE IDEA

Given the limited amount of raw material combined with today's throwaway society, the reuse of goods is of crucial importance. Hence, the disassembly of products should become an essential part of their life-cycle. [1] Therefore, the idea of Disbotics - which is a fancy name for disassembly robotics - is to develop a concept for disassembly of components with mobile autonomous robots. Autonomous robots are robots that can perform desired tasks in unstructured environments without continuous human guidance. [5]

Also, children should be encouraged to research in the field of autonomous robotics in a fun way. By participating in projects and programs like “Botball” [9], young scientists are able to recognize and learn easily about the problems of robotics in disassembly processes.

KNOWLEDGE

A production usually possesses sufficient knowledge about the manufacturing process and the involved machines as well as product parts. However, regarding disassembly processes this Know-how is non-existent. Therefore, to be able to disassemble existing products efficiently, the corresponding components' structure has to be perceived and determined e.g. by using a knowledge base. Optical tracking in combination with knowledge bases and the ability of mobile robots to learn and act autonomously promises to be a solution of the problem of autonomous distributed disassembly. [1]

DISBOTICS - OUR PROJECT

Disbotics is accomplished in cooperation with the Vienna Institute of Technology, the ACIN Institute of the Vienna University of Technology and the KISS Institute for Practical Robotics. [2]

The project was partly done in the course of our project management classes at the Vienna Institute of Technology. Therefore, the following documents were required before the project could be executed:

- Product requirements document:
  defines the requirements for our challenge to disassemble LEGO-Blocks.
- Feasibility study:
  checks the technical feasibility of the project
- Functional specification document:
  documents the technical concept

GOALS

We simplified the issues of autonomic disassembly and decided to take apart simple structures like LEGO-Duplo-Blocks and sort individual parts afterwards on the basis of a specific criterion - in our case color. The purpose of this approach was to win significant core knowledge regarding disassembly processes with mobile robots, while leaving out specific issues and disturbances.

New technologies like Multi-Agent and rule-based systems lead to a decentralized concept with artificial intelligence and promise robustness against disturbances. The usage of semantics copes with the challenges of communication between heterogeneous systems. [1]

DISBOTICS CORE – THE FRAMEWORK

This framework is the fundamental of our research project which is further described in the following chapter. The first version was developed at the Vienna Institute of Technology in the informatics department during the fifth grade in 2010/2011. Since version 1.1, this framework was developed further at the Vienna University of Technology by Clemens Koza and Christoph Krofitsch. [2]

FUNDAMENTALS

The framework "Disbotics Core" was the first step in this major project. A robot controller provided by KIPR, running a Java VM adapted for limited devices is necessary. This framework provides an agent- and rule based programming environment (see description below) with the usage of an ontology, optimized for this limited environment. The ontology supports high-level tasks by allowing the user to create new classes which can be included in several rules to control the robot's behavior. Moreover it also maps robot-specific parts, such as controller, sensors, etc., which can be accessed with rules. Among other things, the user has the possibility to extend the ontology, set rules for it and implement new logic in form of agents. In the current version, the implemented agent-based system is JADE (Java Agent DEvelopment Framework) [10], the used rule based system is OPSJ and the OWL-standard is used for the ontology. The framework is designed with flexibility in mind, so that the multi-agent system, rule engine and hardware (controller) may be exchanged with moderate effort, based on a plugin system. [2]

FUNCTIONALITY

Agent-Based Programming

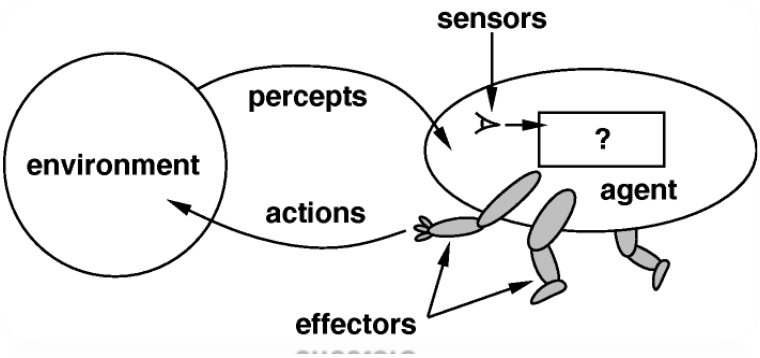

FIGURE 1: THE INTRODUCTION OF AN AGENT

A software-agent is an autonomous unit which uses specific domain knowledge and standardized interaction to solve various tasks.

The question mark represents the knowledge, which interprets perceptions and triggers actions. In our case, the ontology is the knowledge and the rule engine controls the robot's behavior.

Even though figure no. 2 shows the usage of multiple robot controllers, the whole environment is based on one single ontology, which is copied to each of the controllers. So each controller has components of other controllers (e.g. sensors) in its ontology, but only updates its own components.

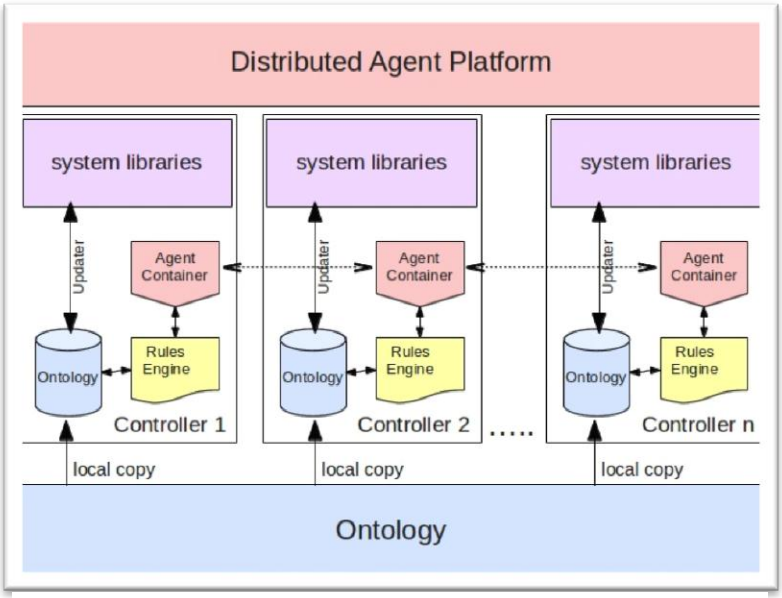

FIGURE 2: BASIC OVERVIEW OF THE FRAMEWORK DISBOTICS CORE [11]

The updating process is performed between the ontology and the specific system-libraries of the controller; in case of the CBC Controller, the CBCJVM is used for that. In this process, the own components mapped in the ontology are synchronized with the values from the system-libraries in order for rules to fire. Each controller has its own rules file, which contains the main behavior of the robot defined in a rule based programming language, such as OPSJ. [2]

The immense benefit of having a global ontology with components of all controllers is the possibility to write rules for components of other controllers. However, these components don't get updated automatically, so it's necessary to communicate with agents of other controllers and request their ontology values. This can be done with specific commands, which are simply invoked in the rules file.

```
import disbotics.core.rules.opsj.DisboticsRuleSet;
import disbotics.core.hardware.cbc.model.Digital;
import model.Flags;

public class rules extends DisboticsRuleSet {
     rule print
     if {
          d: Digital (   d.isValue() == true,
          test equals(d.getNAME(),"controller1_blackbutton"));
     } do {
          command("ont-update", "Grabber", "flag.grab");
     }
}
```

OPSJ is very similar to Java. All used classes have to be imported, and every rule is a class on its own which can inherit from other classes like the DisboticsRuleSet. The rule definition starts with the keyword **rule** which is followed by the name of the rule. Every rule is built like a simple if-then condition whereby the single clauses in the if-condition are automatically connected with a lo1gical AND. So the following example checks if any Digital-sensor has the value true AND if that Digital sensor is named controller1_blackbutton. Only if both statements are true, the rule will fire.

IMPLEMENTATION

CONSTRUCTION

We were committed to construct the robots as compact and stable as possible. Both robots, Grabber and Lifter, are briefly explained in this section, which highlights specific details we thought were worth mentioning.

GRABBER

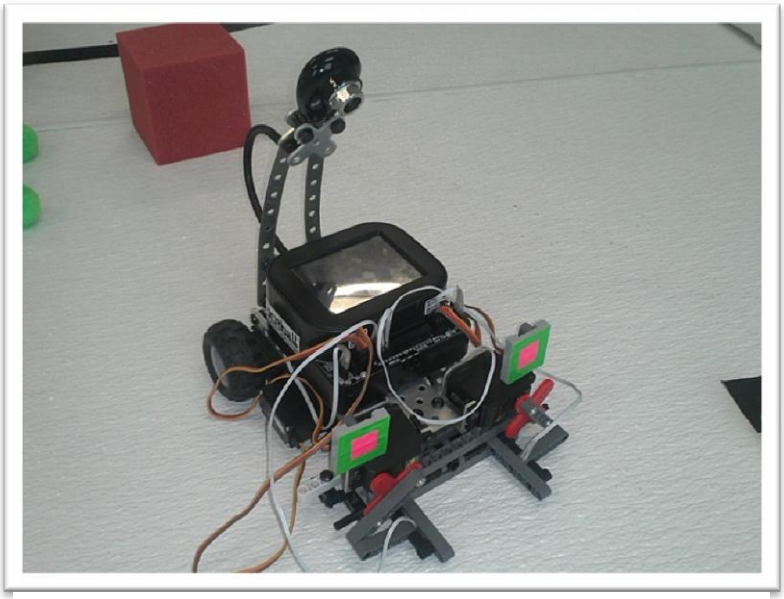

FIGURE 3: CONSTRUCTION GRABBER

This is what the Grabber roughly looks like. The green/red colored blocks were mounted for orientation purposes, so the Grabber can clearly be identified by the Lifter.

Also the Grabber is responsible for sorting individual parts which includes lifting the robot with a lever. By doing that the Grabber can simply place individual parts into suitable containers.

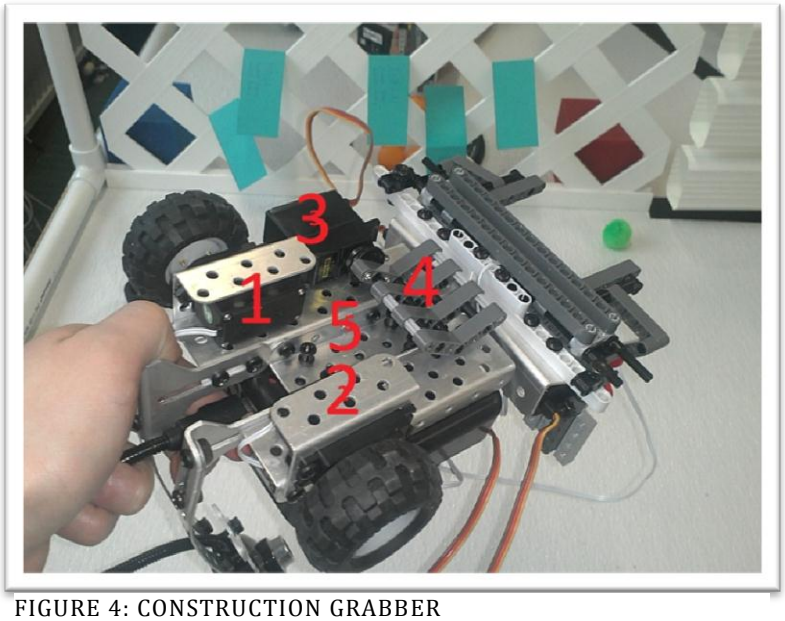

FIGURE 4: CONSTRUCTION GRABBER

The robot has got two ET sensors and five infrared sensors, which are both different distance sensors.

1, 2 … motors to drive
3 … servo to lift the robot
4 … lever (is lifted by the servo)
5 … skeleton of the robot

LIFTER

Basically the Lifter has the same skeleton and interfaces as the Grabber. However, it additionally possesses the function to lift his arms synchronously. These arms can be tilted to the left and the right, by twisting the wheel on which the arms are attached. The foregoing tasks are vital in the process of taking the LEGO-Block apart.

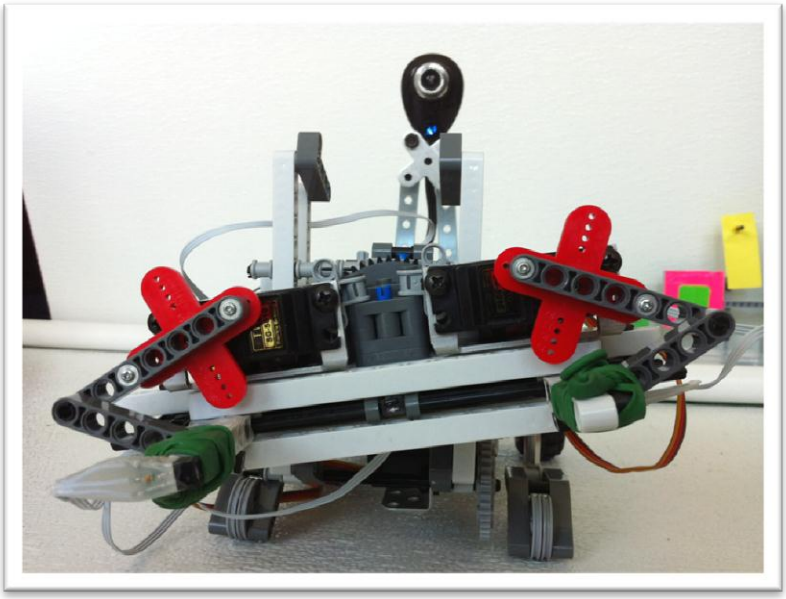

FIGURE 5: CONSTRUCTION LIFTER

Both arms can slide along the rail independent from each other.

The grip of the arms is crucial while holding the LEGO-stone because of the actual disassembly process. Therefore the arms are covered in rubber.

PROGRAMMING

Since the very beginning the programs were divided into high-level and low-level, whereby high level is implemented in the ontology and the rule engine - in our case, OPSJ. Low-level includes methods which can simply be invoked in order to perform predefined sets of action, like driving or picking up things.

HIGH LEVEL

Both robots, Grabber and Lifter, communicate with each other by toggling various variable values after certain actions have been performed.

The communication is implemented in a rules file, which contains a rule for every action, invoking low-level methods and handling other overhead tasks.

Most people would expect that a large part of the tasks and problems are solved by using rules, as this would make the robots much more flexible and intelligent. Even though this is true, reading single sensor values in the ontology and reacting to them by triggering rules turned out to be very slow and impractical. Therefore, we only implemented rules for the communication of both robots, and major steps, when inevitable.

LOW LEVEL

Parallel alignment

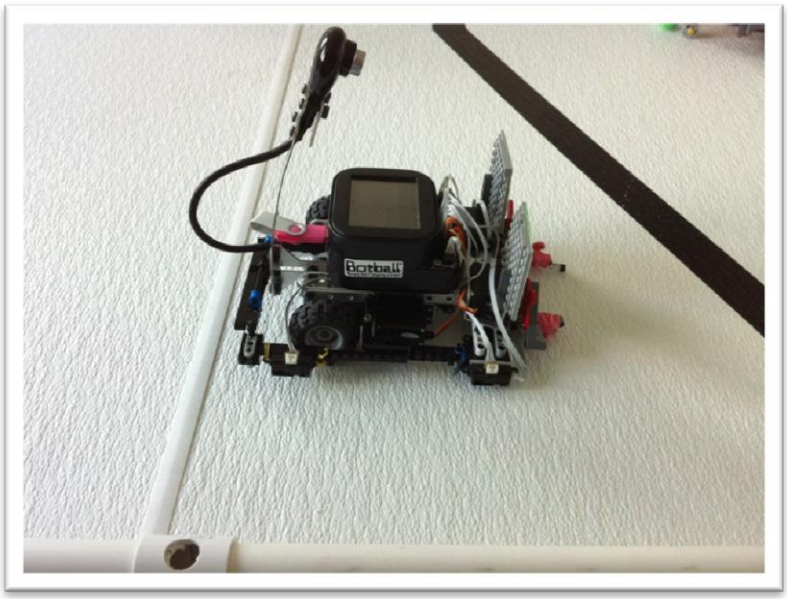

FIGURE 6: PARALLEL ALIGNMENT

There were several orientation tasks to be solved, like positioning the Grabber parallel to the margin of the field. For this purpose, we used 2 ET sensors, which measure the distance to the next hurdle or margin. The sensors were built into each end of a long LEGO-stone. By using a high-pass-filter and comparing both values, the robot could be aligned parallel to the margin of the field.

Artificial color

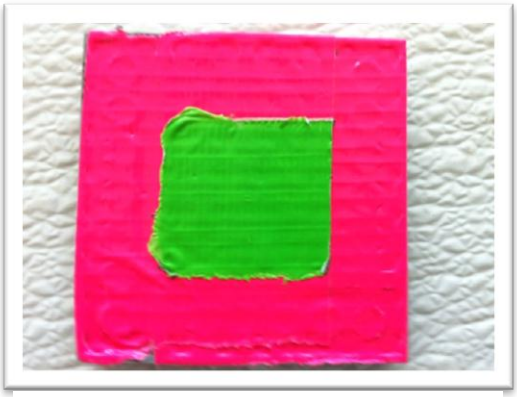

FIGURE 7: ARTIFICIAL COLOR BLOCKS

Basically, the camera only recognizes four colors, which are way too few. However, colors can be "mixed" to artificially "produce" more colors, which can be recognized by the camera. By encircling one color by another we created a specific criterion which can be checked in low-level programs. We handled various orientation tasks by using artificial colors like the ones in the figure no. 7.

"Feeling" arms

As the camera was not reliable enough to determine whether the arms were holding a Lego-stone or not, we used Top-head sensors, which were built into the arms.

LESSONS LEARNED

MINIMIZE POSITIONING

The execution of the disassembly process was planned to be done by 3 robots, whereby each robot would have been responsible for one task like grabbing, taking apart (lifting) and sorting. Also, the process was divided into two fragments:

- Disassembly process
- Mobile path planning / Orientation

Despite our expectations, the disassembly processes were not as complicated to solve as orientation tasks, when working with mobile robots.

The positioning of robots to each other had to be done very exactly in order for the disassembly process to work at all. Therefore it was a logical conclusion to summarize tasks like grabbing and sorting, which could also be done by the same robot and eliminate the third robot. By doing that, we were able to minimize positioning and movement, simplifying the process once again and therefore eliminating potential errors.

CONCLUSION

The issues of autonomic disassembly processes with mobile robots were simplified by working with simple structures like LEGO-Duplo-Blocks. Technologies like multi-agent and rule-based systems were applied because they were considered suitable.
A better camera would allow the robots to be more reactive and therefore more efficient. Therefore this sector could be improved by introducing a better camera.

GLOSSARY

Ontology	In computer science and information science, an ontology formally represents knowledge as a set of concepts, and the relationships between those concepts. [6]
Framework	A software framework is a universal, reusable software platform used to develop applications, products and solutions. Software Frameworks include support programs, compilers, code libraries, an application programming interface (API) and tool sets that bring together all the different components to enable development of a project or solution. [7]
Rule-based system	In computer science, rule-based systems are used as a way to store and manipulate knowledge to interpret information in a useful way. They are often used in artificial intelligence applications and research. [8]

REFERENCES

[1] DI (FH) Mag. Gottfried Koppensteiner;
Brief description of DISBOTICS – Disassembly Robotics;
available at http://www.sparklingscience.at/en/projekte/495-disbotics-disassembly-robotics;
last accessed on 12.04.2012

[2] Clemens Koza, Christoph Krofitsch, Manuel Parg, Prvulovic Antonio;
Disbotics Core Framework user guide;
last accessed on 12.04.2012

[5] Wikipedia article about Autonomous robots;
available at http://en.wikipedia.org/wiki/Autonomous_robot;
last accessed on 16.04.2012

[6] Gruber T.;
The term Ontology was explained on the Stanford educational website;
1992;
available at http://www-ksl.stanford.edu/kst/what-is-an-ontology.html;
last accessed on 16.04.2012

[7] DocForge (An Open Wiki For Software Developers) explains the term 'Framework';
available at http://docforge.com/wiki/Framework;
last accessed on 16.04.2012

[8] Jocelyn Ireson-Paine;
Rule-based systems are explained on the following website for students;
1996;
available at http://www.j-paine.org/students/lectures/lect3/node5.html;
last accessed on 22.04.2012

[9] KISS Institute of Practical Robotics, The Botball Season;
available at http://www.botball.org;
last accessed on 24.05.2012

[10] Telecom Italia Labs, JADE - Java Agent Development Framework;
available at http://jade.tilab.com/;
last accessed on 24.05.2012

[11] Gottfried Koppensteiner, Clemens Koza, Christoph Krofitsch, Manuel Parg, Antonio Prvulovic and Munir Merdan;
Knowledge Based Agent Architecture for High Level Control of Mobile Robots;
presented at the Global Conference on Educational Robotics on July 2011