Simple Robotics with Fujaba

Ruben Jubeh, Albert Zündorf
University of Kassel, Software Engineering,
Department of Computer Science and Electrical Engineering
Wilhelmshöher Allee 73
34121 Kassel, Germany
[ruben | zuendorf]@cs.uni-kassel.de
http://www.se.eecs.uni-kassel.de/se/

ABSTRACT

Fujaba is not only used for professional software development but also for teaching software development. We have used Fujaba successfully in our education efforts for a long time. In order to give students a better experience and feedback, we are preparing programming exercises with a simple robotic system. We are using the Lego Minstorms NXT robotics kit for that purpose. It consists of a micro-controller, motors, several sensors and of course, Lego parts to build different kinds of robots. The micro-controller can be either remote controlled by a PC or run deployed software. This paper describes how we use Fujaba to develop a software which controls a simple forklift robot, which solves the Towers of Hanoi game.

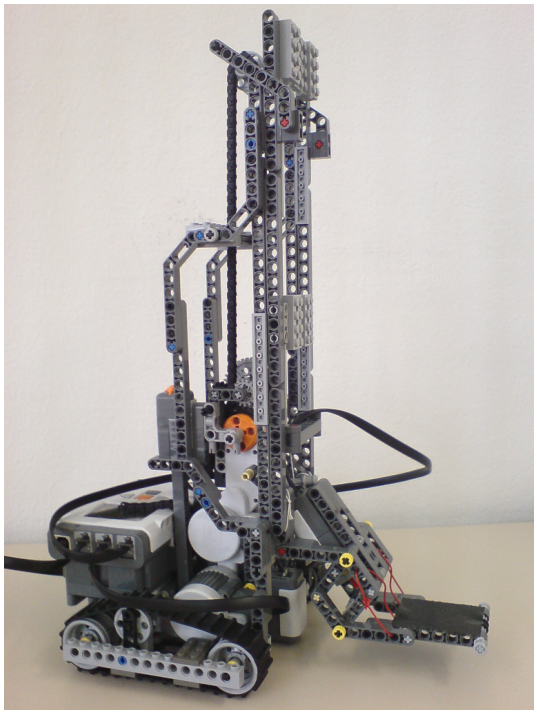

Figure 1: Forklift robot

1. INTRODUCTION

Fujaba is used for teaching software development successfully at Kassel University. In cooperation with the Gaußschule Braunschweig we developed a robot programming course with the predecessor product from Lego, the Mindstorms Robotics Invention System, which was released in 1998. Although it was possible to control a Lego forklift robot with Fujaba, the project faced lots of difficulties and obstacles. For example, the communication between the host PC and the mindstorms microcontroller was done by a infrared link, which was slow and unreliable. Furthermore, it was difficult to have more than one system in a room. More details can be found in [2]. Now we tried to use the newer Lego Mindstorms NXT robotics system [4], released in 2006. It provides improved hardware: a full-featured 32bit microprocessor, step counter motors, graphical LCD display and USB and bluetooth connectivity. The heart of the system, the so called NXT brick containing the micro-controller, can be either remote controlled by a PC or run deployed software. Four sensors and three motors can be connected to the brick.

This paper describes how we use Fujaba to develop a software which controls a simple forklift robot. Section 2 describes how we designed our forklift robot.

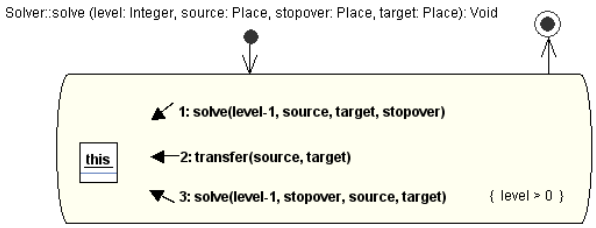

Figure 2: solve()-Method of Hanoi

At the practical side, our forklift robot should solve the Towers of Hanoi game. This is a well documented mathematical game or puzzle which consists of three rods, and a number

of disks of different sizes which can slide onto any rod. The puzzle starts with the disks stacked in order of size on one rod, the smallest at the top, thus making a conical shape. The objective of the puzzle is to move the entire stack step by step to another rod. Only one disk may be moved at a time. Each move consists of taking the top-most disk from one of the rods and sliding it onto another rod, on top of the other disks that may already be present on that rod. No disk may be placed on top of a smaller disk. In our approach, we replaced the disks by wooden blocks and don't use any rods. The blocks just get piled directly onto each other. The recursive software solution is very simple, see Figure 2. The *solve()*-method gets called recursively, *transfer()* simply moves the top disc from the source to the target stack. [1] discusses this recursive solution in detail. To carry that software solution into the real world, each time a disc is moved, the robot should drive to that certain block, grab it, drive to the destination and discard the block.

To keep the control software separately from the concrete problem, we decided to implement it as software library. Figure 3 shows the class diagram of our *FujabaNxtLib*, which is a Fujaba project. The central class is *FNXT* which represents the NXT Brick. The Motor class and the several kinds of Sensor classes are self explanatory. Each sensor class has a corresponding listener interface (not shown). The class *FNavigator* controls the two driving motors and provides methods for moving and turning the robot.

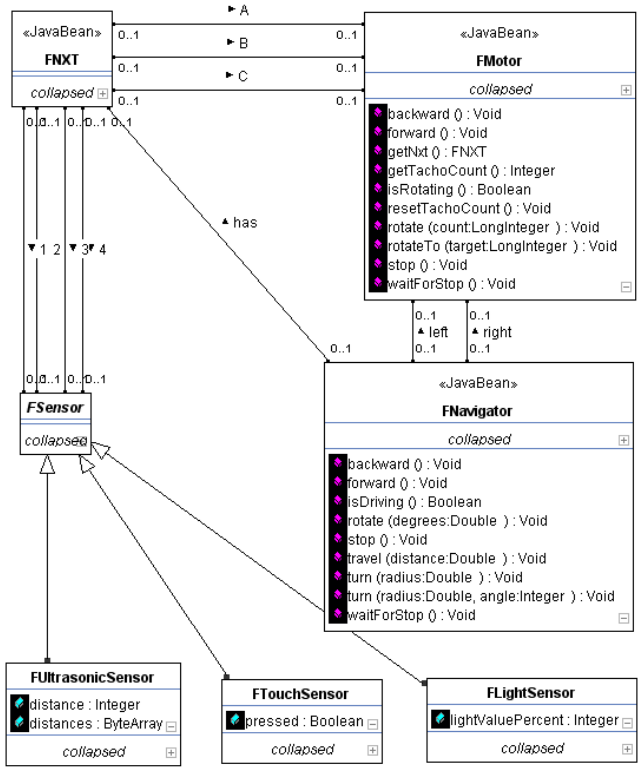

Figure 3: Class Diagram of the Fujaba NXT Library

2. THE ROBOT HARDWARE

It turned out that building the forklift itself was quite difficult. Our early Lego models did not turn properly, were unstable or had a bad center of gravity. Figure 1 shows our current forklift robot. We decided to use two chains instead of wheels so the robot rotates within its footprint. This way, the software may easily predict the turning, which is necessary for exact navigation. The third motor actuates the fork via a long chain and cord. The NXT lego motors contain a step counter: It is possible to turn the motor just a certain amount of rotation, with the accuracy of one degree. So, we don't need a sensor to detect the upper bound for lifting the fork, we use a fixed constant value and synchronize with the ground. Touching the ground resets the step counter. The robot uses only two sensors: a light sensor is mounted just above the ground and can detect changes in the ground contrast. This can be used mark certain places for the robot. The sensor is mounted in the front center of the robot. It can be used to follow the right edge of a black tape line on the ground: When detecting black, the robot turns lightly right while driving forward, otherwise left. The second sensor is mounted on the fork and detects when the fork touches the ground or when it touches an object while driving forward. The trigger mechanism at the fork passes vertical and horizontal force to a single touch sensor. Figure 4 shows the fork sensor mechanism in detail.

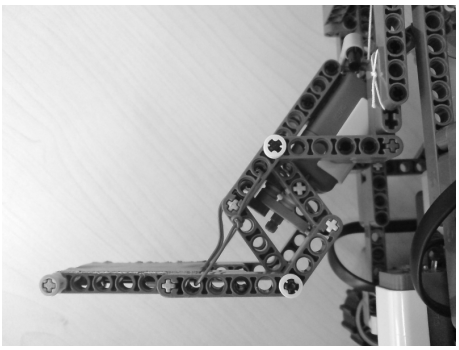

Figure 4: Fork sensor mechanics

3. SOFTWARE ARCHITECTURE

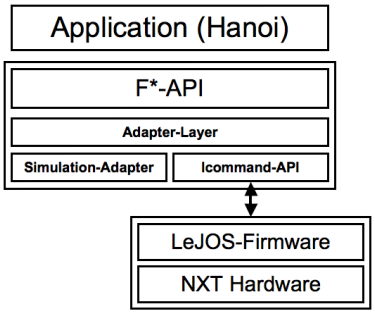

Figure 5: Software architecture layers of the Fujaba/NXT system

We are using the *LeJOS open source java firmware* (see [5]), which includes a Java virtual machine and a Java Micro-Edition-like API. The original Lego firmware has to be replaced by LeJOS. Additionally, LeJOS provides a remote control API called *ICommand* for PCs/Java2SE. We decided to use the latter, and control the NXT remotely over the bluetooth connection with software running on a PC host.

The sensor software components from Lego don't support any listener callbacks on changes directly, it is just possible to ask the sensors for their current value. So, we have to constantly poll all sensors and create events when the sensor values change. This is necessary because the controlling software, which runs on the host, needs to react quickly on events, for example by stopping a motor. Polling the sensors takes only 10-50ms each time, so the polling is performed periodically at a fixed interval. Fortunately, this does not interfere with any control commands. These is a radical improvement over the old Mindstorms system, which had a poll latency of 100-200ms per sensor.

We are planning to replace the poll mechanism by a event notifier running directly on the NXT's micro-controller, which call their listeners by active communication from the brick to the host over the bluetooth connection.

Figure 5 shows the software layers of our architecture. On top is the concrete application model. It uses the F*-API, so classes like *FNXT, FNavigator, FMotor, FSensor* etc. Each F*-Object has a associated adapter instance, which either delegates the calls to the *ICommand API* (Part of the LeJOS package) or simulates the behavior. This is used for testing purposes. The *ICommand API* uses a bluetooth connection and sends simple byte-array commands to the LeJOS firmware on the NXT brick. There the commands are interpreted and mapped to hardware resources.

4. SOFTWARE USAGE AND MODELING

The easiest way to use the Fujaba NXT Library is to start it in debug mode and use eDOBS to create the initial FXNT instance. After that, the methods on the individual motor and sensor instances can be called interactively. Figure 6 shows Eclipse/eDOBS running the library. On the left, you can see the *FNavigator* methods. A call to *travel()* will make the robot drive immediately and block until it finishes.

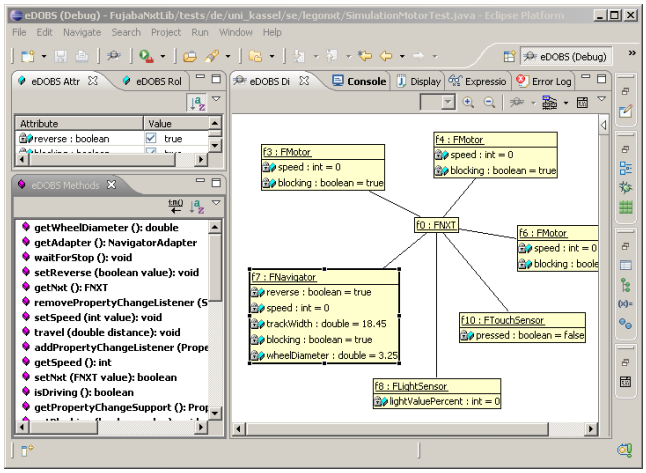

Figure 6: Eclipse/eDOBS running the Fujaba NXT Library

To model the application behavior, both Story or State Diagrams are suitable. To simplify matters, we currently use only story diagrams. Figure 7 shows an example. The method *findInitialPlace()* does not require any control flow, thus it is modeled as just one story activity. However, the single activity itself contains multiple steps and concurrency: some methods (*forward(), backward()*) are asynchronous, that means the method call immediately returns, but the robot acts until a contrary method (*stop()* in that case) is called. After calling a asynchronous method, we use a waitFor-Some-Event-method, in this case *waitForPressed()/ waitForReleased()* on the touch sensor instance. This is a simplification for easy modeling: instead of implementing a listener and being forced to handle the event in some other place of the model, this method blocks until an event of the desired type occurs. This makes reactive programming a bit easier, but semantically means that the single activity is divided into two timing states: the robot drives forward until the sensor hits something, then backwards until the robot is free again.

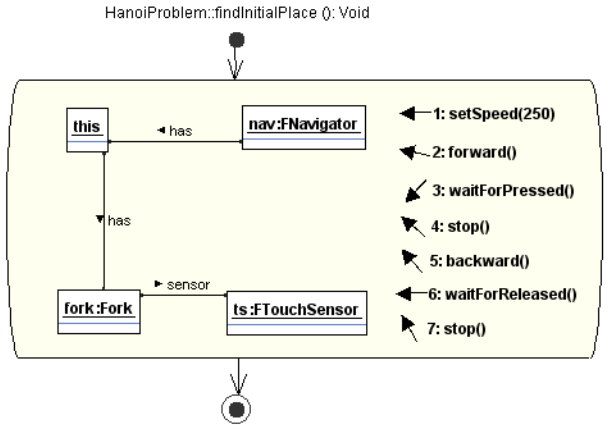

Figure 7: Example of a Hanoi method for finding the blocks

5. PUTTING IT ALL TOGETHER: SOLVING HANOI

Figure 8 shows the class diagram of our Hanoi robot model. It consists of three Fujaba projects: The FujabaNxtLib provides the FNXT, ForkLiftRobot and Fork, thus the abstract modeling to control the hardware. The classes Place, Disc, Hanoi and Solver come from a abstract Hanoi project which can be run or debugged independently from any hardware. The HanoiRobot project, consisting of just HanoiProblem and RobotSolver, depends on the both projects and connects these. We just need to override Solver.transfer() and hook in concrete robot commands to move blocks around there.

Our modeling of the robot-enabled *transfer()* method is just a reference, students will be given only the FujabaNxtLib Library and then should model their own solution.

6. SUMMARY AND FUTURE WORK

We used the new Lego Mindstorms robotics system to implement a forklift robot which can solve the Towers of Hanoi game. The application model was implemented using Fujaba. It is easy to implement the robot control software using standard Fujaba modeling techniques, which makes it suitable for education. By using Fujaba project dependencies, we were able to develop a Library, so modeling control

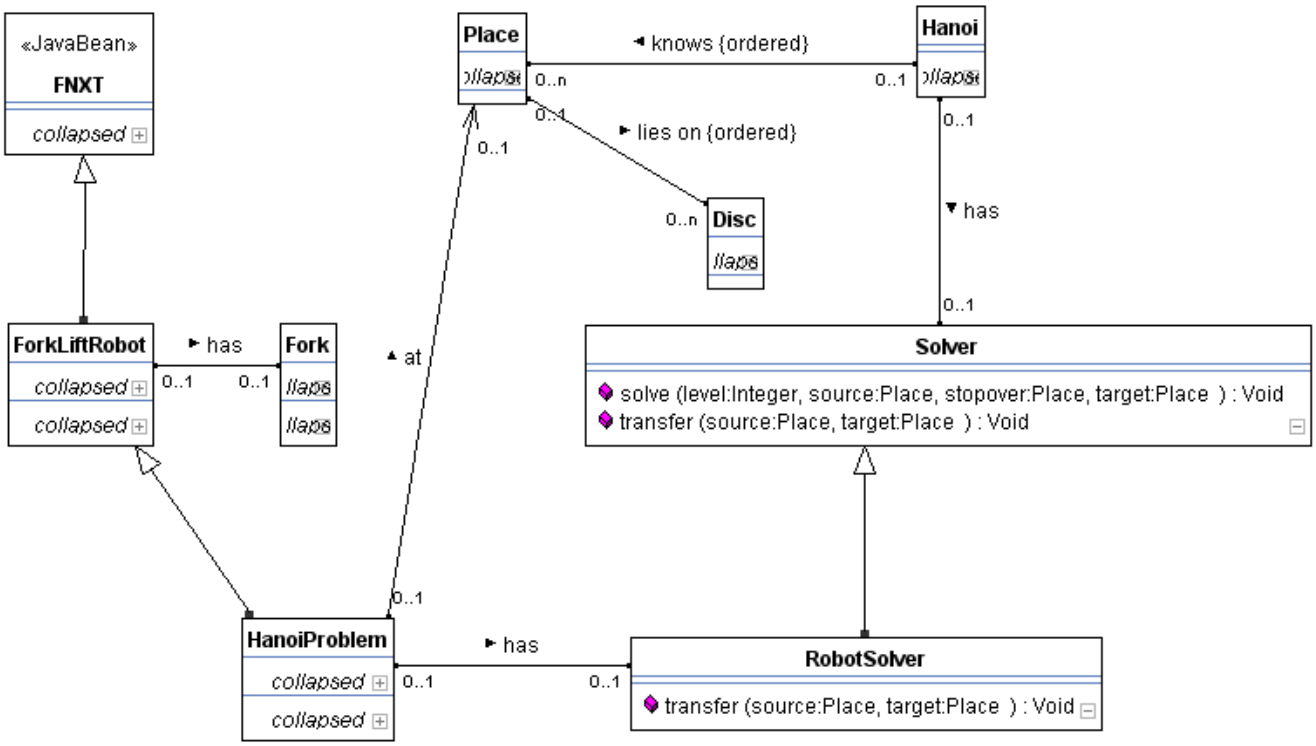

Figure 8: Class diagram of Hanoi, Robot and NXT

software for more applications than just Hanoi is easy. Technically, there are many advantages over the old system: the bluetooth communication is fast and reliable with low latency. On the hardware side, the improved motors with step counters make programming easy as we don't need sensors to detect the motor state anymore. This, and the sensitive fork mechanic free up sensor ports, so the robot can be easily extended with more functionality. We were surprised that the navigation of the robot is quite exact, turning and moving the robot has just a minimal deviation.

We plan to model the forklift as 3D model, so we can simulate it's behavior in a physical environment in software. This should reduce the time for developing and testing software. Due the adapter layer in our software architecture, application models should be able to run in a simulation environment without the need for any changes.

Furthermore, we plan to adapt the Fujaba code generation [3]. This requires a set of templates which consider the limited capabilities of the LeJOS VM, and a runtime library for sensor event handling.

7. REFERENCES

[1] I. Diethelm, L. Geiger, A. Zündorf. UML im Unterricht: Systematische objektorientierte Problemlösung mit Hilfe von Szenarien am Beispiel der Türme von Hanoi. *Erster Workshop der GI-Fachgruppe Didaktik der Informatik, Bommerholz, Germany*, October 2002.

[2] I. Diethelm, L. Geiger, A. Zündorf. Fujaba goes Mindstorms. *Objektorientierte Modellierung zum Anfassen; in Informatik und Schule (INFOS) 2003, München, Germany*, September 2003.

[3] L. Geiger, C. Schneider, C. Record. Template- and modelbased code generation for MDA-Tools. *3rd International Fujaba Days 2005, Paderborn, Germany*, September 2005.

[4] LeJOS, Lego Mindstorms NXT. http://mindstorms.lego.com/, 2008.

[5] LeJOS, Java for Lego Mindstorms. http://lejos.sourceforge.net/, 2008.