Using LEGO MINDSTORMS NXT and LEJOS in an Advanced Software Engineering Course

Michael W. Lew, Thomas B. Horton, Mark S. Sherriff
Department of Computer Science, University of Virginia
{mwlew, tbh3f, sherriff}@virginia.edu

Abstract

This paper describes the benefits of using LeJOS and the Lego Mindstorms NXT set for teaching advanced software development. While Lego Mindstorms has been used in introduction to computer science courses, it is not reported to be widely used in a simulated production environment requiring such things as threading, network communications, and the implementation of command protocols. Additionally, because the Mindstorms NXT system supports Bluetooth communications with multiple devices, it is possible to use this system as the basis for a complex, communicating system requiring multiple software artifacts on different machines.

1. Introduction

The major focus of an advanced software engineering course is to learn how to build software, starting from a set of requirements and ending with a final artifact that satisfies the specifications and is free of defects. In many software engineering projects, this process can be rife with problems, many of which are costly in terms of monetary costs and/or human lives. Examples of such problems have been seen in the Ariane V and Therac 25 disasters. These problems include unclear requirements, building the wrong functionality into the software artifact, poor or non-existent documentation and source code, late delivery, and most critically, execution-time faults.

For example, in the advanced software engineering course at the University of Virginia, the goals of the course center on a collaborative project meant to simulate a customer-developer relationship. A group of four class members serves as a software designer, and the professor and course staff act as a customer who presents the group with a project with specific requirements and milestones. At the end of the course, students should be able to work in a group and across groups to produce a software artifact that is designed well and satisfies the requirements set forth by the professor for a given project [10].

Lego Mindstorms NXT and the LeJOS programming environment is an ideal system for advanced software development courses. The programmable brick supports Bluetooth version 2.0 + EDR communications using the serial port protocol. It is a Class II device, giving a non-line-of-sight communications distance of up to 10 meters, or roughly 30 feet [14]. Because of this, it is possible to use Bluetooth enabled computers, cellular phones, and control devices, like Wii Remotes or Xbox 360 controllers, to control a NXT robot, as well as receive sensor feedback from the motors and attached sensors. This is good for an advanced software engineering course because it enables the assignment of a project requiring communications with other pieces of software, adding complexity to the system and exposing students to programming networked software.

LeJOS uses a commonly-used high-level-language and allows for communications between computers running Java as well as other Lego Mindstorms NXT bricks running LeJOS or the standard Lego firmware [12]. Programs built in Java for a personal computer running the standard Java Runtime Environment can utilize the iCommand API, requiring no program or firmware replacement on the brick, or the LeJOS Java Virtual Machine on the brick and the LeJOS PC Communications library, requiring a program on the brick to receive commands and a controller program on a connected computer [1]. Additionally, mobile phones can be used as controllers by building Java Micro Edition programs for phones that include a Java runtime. Because Java is taught in most introduction to computer science

courses, using a Java-based system imposes no learning curve for a new language, as well as allowing the use of computers with the Java SE runtime and mobile phones running Java ME.

2. Background

In this section, we will discuss the use of robots in CS education and the Lego Mindstorms NXT system in general.

2.1. Robots in CS Education

Commonly, robots are used in introduction to computer science courses. At the George Washington University, the CS 001 introductory class exclusively uses Lego Mindstorms to teach problem solving using computers and applications of computer science to the real world [15]. Likewise, the introductory computer science course at Cansius College uses Lego Mindstorms and the LeJOS system [3]. To teach basic computer science concepts such as control flow and the use of arrays, the United States Air Force Academy has studied using a previous version of Lego Mindstorms to teach an introductory course using Ada [6]. At some colleges, the visual programming language used in the Lego-supplied software to teach the concepts of computer science without making students who may be unfamiliar with a programming language to learn the syntax of a high-level language like C++ or Java, two of the languages that are commonly used in introductory computer science courses [4].

Robots have also been used in single-topic computer science courses. One of the areas that robots are most often used for pedagogical purposes is the field of artificial intelligence. At Williams College, robots are used to introduce students who may not be computer science majors to the concepts of artificial intelligence and behavioral programming through the use of robotic vehicles and a high-level language [5]. The City University of New York, Staten Island, also uses robots to teach artificial intelligence, but the course is tailored for third and fourth year computer science majors and requires a proficiency in C++ [7]. At the University of Salzburg, robots are used to teach embedded software engineering, covering real-time operating systems and communications protocols, scheduling, real-time programming, and code generation [8]. Villanova University has used Mindstorms robots to teach computer organization and operating systems concepts [9].

A common feature in both the introductory courses and the single-topic courses is that none of them require students to design a software artifact from a specification using software engineering practices. In the introductory courses, both for computer science majors and non-computer science majors, the projects are rather simple, like programming a robot to follow a line by using a light sensor, or pushing cans out of a dark area of a board [2, 4]. While it is common for groups to be formed in introductory classes that use robots, the projects do not require, nor do introductory classes teach, software engineering concepts like specifications elucidation, design practices, or verification. Additionally, these classes do not attempt collaboration between groups. In the advanced, single-topic courses, the focus of the course is on a particular concept, and while it requires the use of sound software engineering practices, the course treats knowledge of such practices as a prerequisite [8].

2.2. Lego Mindstorm NXT System

An educational version of the Lego Mindstorms NXT system is available through Lego Education. It contains a programmable brick, three motors, one light sensor, one ultrasonic (sonar) sensor, one sound sensor, and two touch sensors. The programmable brick contains three output ports used to power the motors, and four input ports used to connect sensors. Legacy support is included via converter wires that connect to the NXT programmable brick and legacy 9V motors and Robotics Invention System sensors. Optional additions to the kit include aftermarket sensors from Mindsensors or HiTechnic, which include vision systems, infrared detectors, color sensors, and compass units, among others [1].

The Mindstorms NXT brick uses a 32-bit ARM processor as its main processor, with 256 kilobytes of flash memory available for program storage and 64 kilobytes of RAM for data storage during program execution. To acquire data from the input sensors, another processor is included that has 4 kilobytes of flash memory and 512 bytes of RAM. Bluetooth Class II V2.0 communications are included for wireless connectivity, and a brick can be connected to a USB port on a computer. The motors included in the kit have built-in rotation sensors accurate to +/- 1 degree, and two motors can be synchronized as a drive unit. To give the robot the ability to "see," the ultrasonic sensor, which is accurate to 3 centimeters and can measure up to 255 centimeters, and the light sensor, which can distinguish between light and dark, can be attached to the brick. A sound sensor that can be adjusted to the sensitivity of the human ear can be used to give the robot the ability to hear and react, if programmed, to noises. Finally, the two touch sensors give the ability for a robot to determine if it has been pressed, released, or bumped, and react accordingly [1].

As a replacement for the standard Lego firmware, the LeJOS project has adapted the TinyVM Java Virtual Machine from the original Mindstorms RCX brick to the new, more powerful NXT brick. It has support for threading, arrays, recursion, synchronization, exceptions, non-generic data structures, standard data types, and input and output [13]. The LeJOS virtual machine supports much of the java.util package, but the data structures require that data be stored as type Object and then cast to a type that inherits Object. For input and output, both streams and sockets are available for use. Streams in the java.io package can be used across a Bluetooth serial port connection and require an input and output stream on both the brick and connected device. For control purposes, the LeJOS platform supports the direct connection of Bluetooth-enabled GPS units for spatial location information and keyboards for the navigational control of a robot [12].

3. LeJOS/Mindstorm Project at UVa

In Spring 2009, we used LeJOS and Mindstorms for the first time in our Advanced Software Development course. The semester was split into two projects, called the Minor Project and the Major Project. The Minor Project was intentionally better specified and somewhat easier than the Major Project so that students could first gain an understanding of programming for the Mindstorm system before we required them to do more requirements and design documentation. Another reason we feel that focusing on the technology earlier was better was that they gained an appreciation for the benefits of proper documentation as some parts of their assignment went smoother at the end of the semester.

Both projects revolved around a story about a mission to Mars. During the mission, the escape pod of the spacecraft crashed into a crater. The Minor Project had students building a robot with a winch that could lift the escape pod out of the crater. The robot itself had to be controlled with the accelerometer on a Wii Remote. After the rescue mission, the student teams were informed that the crashed astronauts had uncovered an alien civilization and they had to build a robot that could explore the area autonomously. Further, two teams would have to build their robots such that they could communicate with each other and help direct the other robot. The first task was for one robot to navigate a maze and then send the directions to the other robot. Then the robots had to use a sound sensor to hone in on a distress beacon. Finally, the robots had to be driven to their final destination. Example pictures of the tasks are shown in Figure 1.

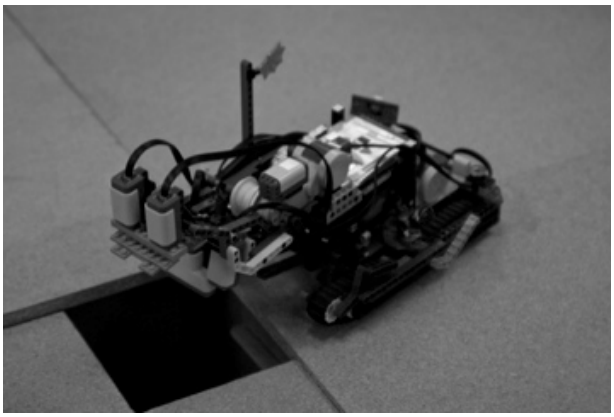
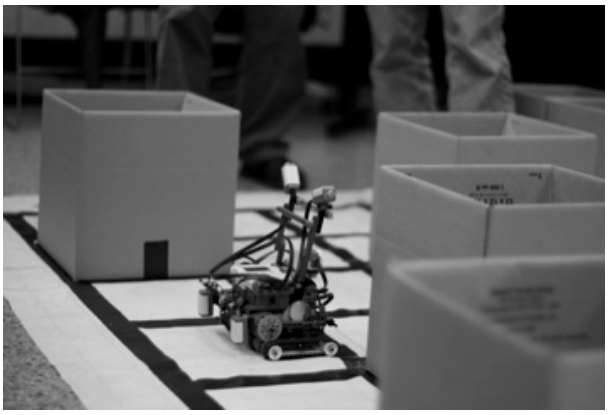

Figure 1. A robot searching for the escape pod and a robot navigating a maze autonomously

The components of the Major and Minor project encompassed numerous technological challenges. Students had to learn Bluetooth for robot and Wii Remote communication. The Wii Remote's accelerometer had to be tied into the system for navigation. AI concepts had to be employed so that the robots could autonomously navigate a maze. The two robots communicated through the host computers over the school's wireless network.

4. Pedagogical Goals

The goals of the advanced software engineering course at the University of Virginia are to introduce students to the practice of software engineering with a simulation of a real-world situation. Specifically, our course goals are for students to:

1. Develop an understanding of how to specify, design, and implement a complex software entity that involves many aspects of modern software systems.
2. Master a number of modern tools and a number of difficult technical fields including user interface design, distributed architecture development, and concurrent programming.
3. Develop the skills necessary to prepare professional quality technical documents as are required in the creation of complex software systems.
4. Develop the presentation skills necessary to convey elaborate and detailed technical concepts to colleagues.
5. Acquire experience of working on a large software system as a member of a group working on system development and as a member of a group that has to interact with other groups and customer representatives.

In general, students are broken up into groups of four to five members, and each group acts as a company. The groups receive a project specification from the course staff, and they have certain milestones that must be met throughout the development timetable. Additionally, a small portion of the course is dedicated to teaching students about the different methods used in software development.

For the Mindstorm projects, we used the Minor and the Major projects to address different aspects of these goals. The Minor project was focused on goals 1 and 2. During the Minor project, which took the first month of class, we did not focus on documentation to any great degree. We did this intentionally to give students a greater opportunity to experiment with the programming tools and technology and to become more familiar with what they could do with it. The students did, however, have to provide basic architecture sketches for their systems.

Concurrently, the course lecture sessions focused on teaching students the methods of software development, emphasizing the steps that should be followed for good software to be developed. As we moved into the Major project in the second half of the course, students were more comfortable with the technology and we began having them write more complete requirements and design documents and give

presentations on their development efforts to their classmates. Further, as teams were forced to work together to build a combined system, the need for thorough documentation became more apparent for the students. Software requirements specification documents were swapped between partner groups so as to simulate receiving requirements from a customer for which software must be built.

The last phase of the course focuses on verification of the software artifacts to ensure that the right functionality was built into the software and that the software works. To validate that the correct functionality was built into the software, groups had to consult the software requirements issued by the customer earlier in the course and verify that each requirement was satisfied. Verification of correct functionality can be done through testing the software artifacts. The system is built correctly if it can complete the trial run within the parameters of the software specifications, and it is easily visible that a system is correct or incorrect based off of the actions taken when, or if, the robot is under autonomous control.

5. Evaluation

The use of Lego Mindstorms and the LeJOS for advanced software development is intended to introduce students to building complex systems utilizing communications and threading. A student survey was done to determine the usefulness of Lego Mindstorms and LeJOS for teaching threading, communications and communications protocols, and building complex systems bringing the concepts of threading and communications together. Specifically, respondents were asked to rate the usefulness of the Lego Mindstorms and LeJOS systems for teaching said concepts on a scale of 1-5. Thirty-nine responses were received, the results of which are summarized in Table 1 below.

Table 1. Results of student survey.

Question	Average score	Pct. of positive responses
(1) Was the major project [a complex project similar to that described above] an interesting or motivating project? (1 = not motivating; 5 = very motivating)	3.92	77
(2) Usefulness of the Lego Mindstorms NXT and LeJOS systems in learning how to build threaded software (1 = not useful; 5 = very useful)	3.52	61
(3) Usefulness of the Lego Mindstorms NXT and LeJOS systems in learning how to plan and implement custom communications protocols	3.53	63
(4) Usefulness of the Lego Mindstorms NXT and LeJOS systems in learning how to build a complex system involving threading and communications between multiple devices	3.71	71

Students were also asked about problems encountered in implementing communications protocols with LeJOS. Many of the problems centered on Bluetooth communications, with students finding that it was easy to overload the brick, that the Bluetooth connection could be unpredictable, and that LeJOS does not support the full functionality of the String type or sending Strings over a Bluetooth connection.

Based on the student response, it would seem that, overall, using Lego Mindstorms and LeJOS to teach advanced software engineering was somewhat useful. On Question 1, 77 percent of respondents believed that the major project was a somewhat or very motivating project. Regarding the usefulness of the Lego Mindstorms system and LeJOS in learning how to build threaded software, Question 2 of the survey, 61 percent of respondents who implemented threading indicated that the systems used were somewhat or very useful in learning the concept. The response to Question 3, the usefulness of the system in learning about communications and communications protocols, indicated that 63 percent of respondents viewed the Lego Mindstorms system and LeJOS as a somewhat or very useful tool for learning said concept. Finally, 71 percent of respondents believed that the Lego Mindstorms and LeJOS systems were a somewhat or very useful platform for learning about building complex systems involving threading and

communications between multiple devices. Overall, based on the average of the responses, students believed that the Lego Mindstorms and LeJOS systems were somewhat useful in learning about software development.

Judging from course evaluations, it seems that a number of students were frustrated with the Bluetooth libraries used to communicate between devices like the Wii Remote and the NXT brick, as well as with problems getting their Bluetooth dongle to work with the Wii Remote, NXT brick, or both. Also, some students felt that working with Lego Mindstorms presented frustrations with compatibility problems with various computers and was too hardware-focused for a software engineering course. It is likely that the students who experienced the most problems with the hardware used for the project also did not think that the project was useful for achieving the goals of an advanced software development course.

The instructor of the course (the third author of this paper) was also asked to rate the effectiveness of the Lego Mindstorms system and LeJOS for teaching advanced software development. The Lego Mindstorms system was rated as neutral on teaching threading, somewhat useful on teaching communications and communications protocols, and somewhat useful in teaching software engineering techniques. Overall, Lego Mindstorms and LeJOS was rated as a good system for teaching advanced software development, despite the fact that the code base was too small for certain software engineering aspects, like subversion and code analysis, to be as useful as it could have been. From a technical standpoint, difficulty with Bluetooth communications was singled out as a particular problem.

6. Other Possible Assignments and Projects Using LeJOS

We have discussed in detail how we made use of Mindstorms and LeJOS in our course's project. But there are many possible ways software engineering instructors could incorporate this robot environment into their courses. In this section, we outline a number of possible assignments and projects.

LeJOS supports both Bluetooth wireless communications through the serial port protocol as well as communications over a USB connection. In the minor assignment addressing communications, it is not required that threading be implemented, as this is designed to be an overview to the process by which a computer or other remote device connects to an NXT brick, sends data, and receives data. Regardless of whether Bluetooth or USB is used to connect to an NXT brick, the connection will implement a DataInputStream and a DataOutputStream generated when a connection is opened between an NXT brick and a remote device.

A simple communications project is included in the LeJOS distribution in the [installation path]\samples\ directory. It consists of two projects – BTSend, which runs on a PC and establishes a connection with an NXT brick, and BTReceive, which runs on an NXT brick and waits for a connection to be made from a remote device. The PC sends an integer to the NXT brick, which negates that integer and sends it back. This project does not require threading because the data input stream is not required to wait a long time for data to be flushed through the data output stream. Concurrent communications is not implemented in this; however, a variation on this may be made to implement threaded communications so that both the PC and NXT brick send and receive integers simultaneously. This would require threading so that both the PC and NXT brick receive data without having to wait for a response from the opposite side of the connection.

Another project that may be assigned is the creation of a communications protocol that defines how commands are sent to an NXT brick and how data is sent from an NXT brick. Because the communications protocol is undefined if iCommand is not allowed to be used, the implementation of a command protocol is a necessity for a major project involving communications between a PC and NXT brick. Using the framework of a thread for input and a thread for output, a command interpreter can be implemented on the NXT brick in the input thread, and periodic data output by requesting data from attached sensors can be implemented in the output thread. This project may be assigned as a collaborative effort between two partner teams, where both teams specify a communications protocol, one team implements the NXT brick side of the system, and the other team implements the PC side of the system. The NXT side of this system would implement a command interpreter and data sender, and the PC side of

the system would implement a data listener and a command sender. To emphasize good software engineering practices, a communications protocol requirements document should be required such that the two teams involved in creating and implementing a communications protocol have firm documentation to consult during development, avoiding conflicts from the wrong commands being implemented in either the interpreter or the command sender.

Because of the limitations of the LeJOS virtual machine in supporting some basic Java methods, such methods as String.split, groups are forced to find alternative methods to accomplish their goal, as can happen in industrial projects. In this particular example, teams could either devise their own string splitting method to handle textual commands or create a numerical command protocol that can be sent using the methods supplied by the DataOutputStream object. Additionally, groups will have to be careful about the amount of data sent to and from the robot over the Bluetooth serial connection, as a high amount of traffic will slow down the reaction time to commands appreciably. While this can be mitigated, to some extent, by the use of threading, a high amount of traffic will cause commands to be buffered in the DataInputStream object, and recently sent commands will not be executed in a timely fashion.

In the scope of a major project involving threaded communications, these projects give each group the knowledge necessary to implement communications in a larger project, both for concurrent input and output as well as sequential input and output. Creating a command interpreter and command protocol is greatly beneficial for the development of the final software artifacts to be developed for the course. Because portions of prior software are reused in newer software, this offers an opportunity for the professor and course staff to teach the proper methods to reuse portions of software artifacts and avoid disasters, like the Ariane V explosion, that result from improper reuse of software artifacts or portions thereof.

Designing and implementing a complex and/or a collaborating system is a task that is a major focus in many advanced software engineering course. For this, robots are ideal because the design and implementation of a complex and collaborating system requires software to be designed to be able to do a series of tasks, not necessarily controlled from a connected device, and send data to a connected device that can be utilized by multiple robots. As noted above, LeJOS can be used to implement such a system through its communications capabilities, ability to program autonomous behavior into a program, and use of Java. Control need not be limited to simple mouse clicks or button presses on a keyboard; Java libraries exist for the use of other control devices connected to a PC.

An additional layer of complexity may be added with the use of the Wii remote as a control device, rather than using a keyboard's arrow keys or a mouse-based interface to control the robot. Instead of using the arrow pad as the primary means of control, the accelerometers used in the Wii remote may be specified as the input from which the robots will run on. By using the Wii remote, students learn how to build software that integrates multiple unfamiliar libraries – the LeJOS and WiiRemoteJ libraries – in a seamless fashion. If use of the accelerometer feature is added as a requirement, it introduces the problem of interpreting data and sending a command, or series of commands, based on that data.

7. Conclusion

Because LeJOS supports networked communications and threading, as well as the fact that a PC-side Java program can be extended to support additional control devices, integrated databases, and other features, it is an ideal platform from which Advanced Software Engineering courses can be taught. Simple projects may be assigned at the beginning to introduce students to the limitations and features of LeJOS and the Lego Mindstorms NXT system. From those simpler projects, students should be able to complete a complex project such as the ones described above with the appropriate documentation and a set of software artifacts with the correct features implemented and verified to work correctly.

Instructors that consider a software engineering course centered on the use of Lego Mindstorms should ensure that enough time and resources are set aside for the upkeep and maintenance of the Lego kits and for technological issues that might arise. Significant time was spent in answering Bluetooth connectivity

questions, which led to some frustration from the students. The difficulties were an opportunity for students to see how technological issues can arise in systems, but the frustration did detract some from the learning experience. Overall, though, there are sufficient technical challenges with the use of Lego Mindstorms that they will continue to be used in this course in future semesters.

8. References

[1] Bagnall, Brian. *Maximum Lego NXT*. Winnipeg, Manitoba: Variant Press, 2007.

[2] Burhans, D. 2007. A robotics introduction to computer science. Paper presented at the AAAI spring symposium, March 26-28, in Stanford, California, USA.

[3] Burhans, D., Meyer, R., VanVerth, P., Puehn, D., Steck, V., & Wiejaczka, J. 2006. Introductory computer science with robots. Paper presented at the proceedings of the 21st National Conference on Artificial Intelligence, July 16-20, in Boston, Massachusetts, USA.

[4] Cliburn, D. 2006. Experiences with the Lego Mindstorms™ through the undergraduate computer science curriculum. Paper presented at the 36th ASEE/IEEE Frontiers in Education conference, October 28-31, 2006, in San Diego, California, USA.

[5] Danyluk, A. 2004. Using robotics to motivate learning in an AI course for non-majors. Paper presented at the AAAI spring symposium, March 22-24, in Stanford, California, USA.

[6] Fagin, B., Merkle, L., & Eggers, T. 2001. Teaching computer science with robotics using Ada/Mindstorms 2.0. Paper presented at the proceedings of the 2001 annual ACM SIGAda international conference on Ada, September 30-October 3, in Bloomington, Minnesota, USA.

[7] Imberman, S. 2003. Teaching neural networks using Lego-Handyboard robots in an artificial intelligence course. Paper presented at the proceedings of the 34th SIGCSE technical symposium on computer science education, February 19-23, in Reno, Nevada, USA.

[8] Kirsch, K. Embedded software engineering course. University of Salzburg. http://cs.uni-salzburg.at/~ck/teaching/EECS290O-Spring-2002

[9] Klassner, F., & Continanza, C. 2007. Mindstorms without robotics: an alternative to simulations in systems courses. Paper presented at the proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education, March 7-1-, in Covington, KY, USA.

[10] Knight, J., & Horton, T. 2005. Evaluating a software engineering project course model based on studio presentations. Paper presented at the proceedings of the 35th ASEE/IEEE Frontiers in Education Conference, October 19-22, 2005, in Indianapolis, IN, USA.

[11] The Lego Group. 2006. Lego Mindstorms NXT Bluetooth Developer Kit. http://cache.lego.com/upload/contentTemplating/MindstormsOverview/otherfiles/2057/LEGO%20MINDSTORMS%20NXT%20Bluetooth%20Developer%20Kit_58CE458E-5292-4CB0-93D2-4BEC821C13C2.zip

[12] LeJOS Team. 2008. LeJOS NXJ API. http://lejos.sourceforge.net/nxt/nxj/api/index.html

[13] LeJOS Team. 2008. NXJ technology. http://lejos.sourceforge.net/nxj.php

[14] LeJOS Team. 2008. The LeJOS NXT tutorial. http://lejos.sourceforge.net/nxt/nxj/tutorial/index.htm

[15] Narahari, B. CS 1: computer science orientation. The George Washington University School of Engineering and Applied Science. http://www.seas.gwu.edu/~bhagiweb/cs1