Creating an Elastic Robotics Framework for Education and Research

Giovanni Vincenti and Goran Trajkovski

Department of Computer and Information Sciences
Towson University
8000 York Rd.
Towson, MD 21204
{gvince1, gtrajkovski}@towson.edu

Abstract

Different applications of robotics require widely different setups of the environments that control the robots. Education and research are closely related in most academic settings yet require different architectures for the structures supporting the operations of agents. Learning how to use a robot needs clear and easy interfaces, while using a robot to discover new knowledge in the field of imitation-based robotics sometimes leaves behind a messy environment. This article introduces a blueprint adaptable enough for students and researchers to survive within the same niche as well as a cheap solution for starting a Robotics laboratory on a shoestring.

Introduction

The products of laboratories of robotics astonish scientists as much as the crowds. Robotics is a sensational field that produces machines that resemble humans, or perform functions that are closely associated with tasks that men and women carry out. Robotics shouldn't be a prize for a few elected researchers, but its multifaceted universe of interdisciplinary concepts lends itself to big projects as much as to the small ones.

In this paper we illustrate how a small robotics laboratory for undergraduate and graduate education at a state university designed a robotics environment flexible and adaptable enough to allow students to learn and at the same time let researchers perform research activities. This framework is based on concepts that blend ideas expressed in Maxwell and Meeden (2000), bringing up the point that research and education must be linked together, when it comes to robotics and its multidisciplinary approaches. Weinberg et al. (2001) also take the same point of view, keeping in mind that not all students are at the same level of understanding about this field when they take their first steps. We would like to add that such environments should be designed also keeping in mind students with higher knowledge and understanding that are transitioning from an introductory level to being part of a team of researchers.

Because so many students have to interact with the lab, the structure needs to be as fault-tolerant as it needs to have potential for growth. At the same time, not all colleges and universities can afford expensive robotics equipment. For this reason there is an increasing need for the creation of relatively cheap solutions that can accommodate learning and research. The recipes for such environments have been discussed by Hsiu et al. (2003), Greenwald and Artz (2004) and finally Trajkovski et al. (2005) to cite a few. The main point that these authors stress is the need for a structure that functions well wearing the many hats that a higher education setting requires.

In this paper we propose an adaptable framework that fulfills multiple needs, from the basic ones of a student who is starting to learn about robotics, to the more complex ones that arise from multiple agents trying to coordinate and navigate through a hostile environment for research purposes. Moreover, this framework will be based on freely available technologies, so that any size institution can get started following this simple recipe.

This paper is organized in the following manner: Section 2 reviews some architectures used in robotics, for purposes of general use and education. Section 3 then analyzes Java and we report our reasons for choosing it to create our framework. Section 4 describes the environment where this framework will have to operate. Section 5 describes the proposed framework. Section 6 describes some of the applications that will use this framework to operate. Finally, section 7 will conclude this article.

Architectures for Robotics

Any type of framework needs a good and flexible architecture. Such architecture must take into consideration the possibility of adapting to a one-robot type of operation, all the way to multiple robots trying to collaborate to carry out some tasks. For this reason, it is important to analyze multiple architectures, before we introduce our solution. We will take into consideration architectures based primarily on Java for reasons we will explain later on in the article.

Kiniry and Zimmerman (1997) review some of the early frameworks that were available on the market. They found that these products include the following commonalities:

• An agent server where agents converge their communications;

• In the presence of multiple agent servers, the agents can move from server to server carrying their state with them;
• The instructions and code that the agents use can be loaded from multiple sources, including the local file system, the web or some ftp service;
• All the frameworks are realized in Java.

The authors also list some frameworks that were available, but were geared towards research projects. We will focus primarily on research- and education-based technologies from now on, but we see that the features highlighted for commercial products also appear in non-commercial ones.

Architectures for generic operations

Multiagency deals, in most cases, with the study of the interaction of robots operating within some common environment. Simmons et al. (2001) highlight the importance of the need for a central location to elaborate plans and coordinate the various robots that are present within one environment. They propose an architecture where each component of this multi-agent system is linked to the main processing center.

Andronache and Scheutz (2005) approach the problem from a different point of view. They create an architecture that transforms a single-agent environment into a multi-agent one. Their initial platform is the APOC architecture (Scheutz and Andronache, 2003a/b), basis that was modified to accommodate for distributed computations as well as a communications infrastructure for agents to exchange information directly.

Pomiers and Dupourqué (2006) add to the concepts of distributed computing and ease of networking between agents the concept of modularity. In according to their paper, in the imminent future there will be the need for agent systems that can host a variety of critical and non-critical modules that will perform a variety of operations. The framework that controls the agents should be ready to accept such modules, whichever they might be.

Architectures for research and operations in general seem to carry the most weight of importance on the integration of multiple agents at the level of ease of communications as well as the ability to distribute processing, still retaining control at a central location. We will now analyze some concepts that are most relevant when the primary aim of the robots is the education of students.

Robotics for education

When students attend an introductory course, the focus of the material should be kept within the boundaries of a single field, as not to bring much confusion to the minds already busy trying to evaluate and incorporate new concepts. The field of robotics, as we already stated, is a field that includes several disciplines. Some of these disciplines can be measured more easily, such as engineering and mechanics, while others are more related to philosophical and social issues, especially when we deal with multiagency.

White et al. (2004) outline a set of concepts that they feel are an important part of every introductory robotics course. These concepts are:
• The course should include hands-on experience in practical robotics;
• The course should teach about integrated system design;
• The students should learn how to interact with people in different disciplines;
• The students should learn about group dynamics and teamwork.

These points are extremely important for a successful course in robotics, but they leave little room for an elaborate and extremely technical framework that will accommodate a research environment. The elements that the students need to learn and that compose the framework need to be part of a higher-level of abstraction that even novices can comprehend, but at the same time they need to include concepts from disciplines that will shape the experience of the students.

Greenwald and Artz (2004) propose a solution to teaching Artificial Intelligence to students by means of cheap robotics solutions. The technical choice for Greenwald and Artz is the LEGO Mindstorms platform. Such platform is easy enough for students to interact with, but at the same time it can become the launch pad for quite in-depth concepts and experiments. The problem that they reveal as predominant in a low-budget solution for their approach is the lack of accurate localization mechanisms. Such component is vital to most agent systems because the agents need to orient themselves in an environment that needs to be internalized every time a new simulation arises. Such need is dictated by the mobility of most classrooms, where backpacks, chairs and objects in general are not permanently situated through a whole semester of classes.

Both of these examples show that there is no real framework associated with education and robotics. They both highlight aspects that are important in an educational environment, and each of these studies takes into consideration only a single-agent type of setting. We would like to extend such concept to accommodate for multiple robots, as to create a more interactive experience for students of robotics.

Java applied to Robotics

Most of the frameworks introduced in the earlier section are realized in Java for the simple reason that Java is a multi-platform language. Because of its nature, the code is written once, and then it runs anywhere. As long as the agent is provided with a Java Virtual Machine (JVM) to interpret the code, the agent will be able to execute the commands specified.

The JVM is available in various editions. This is perhaps the second main advantage for using Java as the foundation to an elaborate architecture. Sun offers the Enterprise Edition (EE), the Standard Edition (SE), and finally the

Micro Edition (ME), as advertised on their website (SDN). These various solutions offer a wide choice for all parts related to a robotics framework, from the servers running the Enterprise Edition to the individual robots that can operate using the Standard or the Micro edition, depending on the architecture of the system.

Java is also a great candidate for these types of operations because of its network-oriented nature (SDN, 2006). As every environment will have to rely on different technologies to carry out messages from one point to another, Java is the common denominator that can link most network architectures with very little effort on the side of the developers.

Wong et al. (1997) give a brief overview of some of the technologies involved with Java applied to robotics at the time when the blend of these two concepts was still young. Java Native Interfaces (JNI or Jini) and Remote Method Invocation (RMI) are the two concepts that appear most frequently in their article.

Jini is the set of the Java language that allows legacy code to be used within a Java program. This part of the language is extremely useful when we are dealing with agents that have been written in different languages, or when we have agents that cannot host a JVM on their embedded system. The interface created with Jini will integrate fully the agent created using a different technology, and will give it a higher-level integration with the rest of the framework.

RMI is also a vital concept to agents and robotic frameworks because RMI allows agents to invoke methods or are granted access directly to the central computing locations of the framework. We allow some freedom for each agent to have its own state. But when the agents need to be coordinated centrally, RMI ensures that each agent will access the very same control thread. This means that the environment on the server will look identical to each of the agent-threads, which will most likely be running at different stages.

Many object that Java is not the perfect solution to every problem. Given the nature of its portability among different architecture platforms, Java has a layer added to the usual concept of "code → runnable program". Each Java program is translated into bytecode. Such bytecode will then need to be translated into machine instructions that the computer will then execute. This action is performed within the Java Virtual Machine, which ensures that the bytecode is translated to the appropriate machine code for the architecture in question. This process is treated extensively in Gagnon (2002). This extra step and the final compilation into a runnable program only at the time of execution through a Just In Time (JIT) compiler delays the prompt start-up of the operations, or the quick response when new functions are added while the core of operations are loading classes dynamically.

Other issues that may arise are discussed in Davison (2005). Although the drawbacks are discussed in the light of creating games in Java, the multiplicity of this programming language allows us to face similar problems. One of the issues discussed is the possibility of memory leaks. Most of the times, this issue is not relevant because of the nature of Java, and its garbage-collection mechanism. Java code is optimized for as few memory leaks as possible. It is also important to note that the garbage collection is argued to execute at somewhat random times, thus making a system with little memory somewhat unreliable. Another issue is the inability to run a Java program on a machine that does not have a Java Virtual machine. Finally, Java is too high-level for applications that demand a quick response and strict control.

We feel that, Java offers enough positive aspects that it should be considered as the language of choice for this project. The negative issues that are part of this programming language can be overcome either through more careful programming or simply with the use of concepts such as Java Native Interfaces, at times when speed and tight control are extremely important, or when a Java Virtual Machine is not available for a certain platform.

The Cognitive Agency and Robotics Lab

The Cognitive Agency and Robotics Laboratory (CARoL) is the outcome of a joint National Academies of the Sciences/NSF venture. This lab plays a pivotal role in the undergraduate and graduate education of Computer Science and Computer Information Systems majors (and non-majors) at Towson University. Used by the majority of students as a setting for their first steps into the world of robotics, CARoL also is home to several undergraduate, graduate and doctoral students who perform research in the areas of adaptable learning systems on simulation and empirical studies. This lab is one of the most active research labs for undergraduates on the Towson University campus. Student research has been presented at a variety of forums, and publications have been published in proceedings of national and international conferences.

The framework that is described in this article will become an integral part of the daily operations of this lab, for tasks that are geared towards education as well as research. As the members of this lab teach all the robotics-related courses to the Computer Science majors and graduate students, on one hand, and the whole Towson University community via general education courses, the results from this project will be integrated in our courses, and used as case study when learning phenomena in humans and in intelligent agents are discussed.

We are in the process of designing a course in multiagent systems on a graduate level, and learning, heterogeneous and homogenous societies of agents will take at least 25% of the topics covered, and projects for students in that class will be directly linked to a wide range of aspects analyzed in this project. This will enable us to test the operations of our framework at multiple levels, from the interaction with students new to robotics to

students with advanced projects as well as researchers performing cutting-edge studies on multiagency.

Creating a framework for CARoL

Given the multiplicity of interests that revolve around CARoL, we propose this framework as the solution that will enable this active laboratory to flourish. Trajkovski et al. (2005) explains in detail all the technologies that are present in this environment. Some of the elements that need to be integrated include LEGO Mindstorms for introductory and advanced courses in robotics, custom-build Hexi robots powered by systems running Palm OS and Windows CE and Pioneer robots by ActivMedia Robotics.

Users

The first issue that needs to be addressed is the method through which users can interact with the framework. Researchers as well as expert students should have unrestricted access to the framework, while novice users should be monitored both in the range of robots they can control and in the operations that these robots can perform. Moreover, some types of experiments may not be immediate, leaving the need for a constant monitoring service that can be operated remotely. Such remote control should not only monitor, but it should also have as much control on the agent(s) as the local interface. This is due to the fact that the robot may have problems, so the user should be able to reprogram its operations or completely stop the simulation at any point.

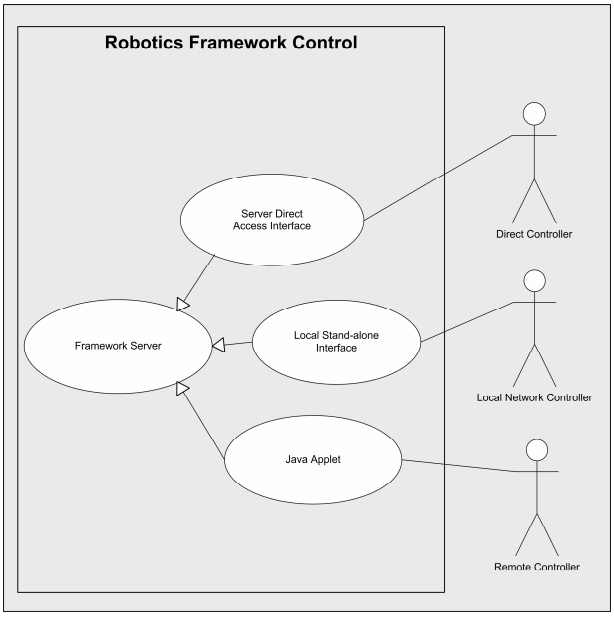

Figure 1: Use case for different users

Figure 1 shows the use case for the three users we can see, Direct Controller, Local Network Controller, and Remote Controller. Novice users would be restricted to access the system from a Local Network interface, while expert users as well as researchers should be able to access the system at any level. A controller who has direct access to the system will have an interface provided by the actual server. Local access can be granted to users by means of a stand-alone interface. Since this interface is the one that would give access to both novice and expert users, there is the need to have an interface with multiple levels of access. Novice users should be able to interact only with LEGO robots, while experts and researchers should have access to the full array of options. Finally, remote users should be able to connect to the framework by means of an applet. An applet is a solution that seems the most viable because a stand-alone application may have limitations due to the setup of the user's machine, but an applet is heavily relying on the Internet browser for any framing issues. Applets are flexible enough to create network connections for live feeds to and from the robot for sending controls and receiving readings from sensors or images from monitoring cameras.

Framework

After defining the types of controllers this framework can host, we can look into the structure of the system, shown in figure 2.

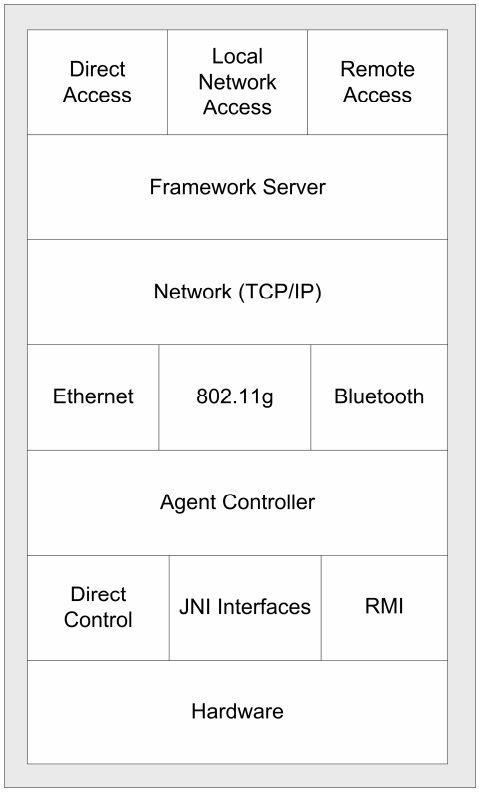

Figure 2: Framework architecture

We already analyzed the top layer of this structure, so we will not spend any more time on it. The second layer, the

Framework Server, is the layer where most of the logic will be stored. Different simulations will take different shape in this layer. The morphology of this layer will change as we deal with single user-single robot interactions in the event of a student learning how to use a LEGO robot, or if we deal with a multi-user-multi-agent simulation that deals with several agents performing functions at once. The Framework Server layer can be implemented using J2EE technologies.

For network communications we will use TCP/IP, given the nature of the environment. We should create a system where packet-loss is not allowed, should one of the robots be in a critical situation where it needs to be reprogrammed or stopped immediately. The TCP/IP layer will then rely on different hardware solutions. Towson University is currently offering a campus-wide Ethernet as well as wireless network featuring 802.11g; the laboratory then implemented a private Bluetooth network that does not extend past the physical boundaries of that area.

The communication will then reach the robot's controller. Such controller will be also created using Java, and it will be responsible for communicating directly with the hardware. Such communication can take place using different methodologies. The first method is through direct control of the hardware. Although this method would be the easiest to implement, it is also the rarest, since there are not many commercially available robots that integrate well with Java. Perhaps the most common method of control will be by means of Java Native Interfaces. This technology will allow us to wrap the original control methods created specifically for the hardware and integrate them with higher-level controllers. Finally, RMI is important for direct control of the robot, in case of failure, or in the event of specific tasks that require global synchronization among all the agents within the system. The bottom layer is the hardware. The robots used in the lab were discussed previously.

In the event of an agent trying to communicate with a second agent, the framework described above will serve just as a regular network. The message sent from one agent to a second agent will travel from the hardware level of the originating point upwards, reaching the Network level at the server, or connecting directly with the receiving agent, depending on the technology in use. After the connection is established, the sending agent will send the information to the receiving one, thus achieving a relative independence from the server when inter-agent communications are required.

Since orientation is one of the issues that were discussed by Greenwald and Artz (2004), we found that also in our solution, given the budget constraints, this concept will be an issue. To solve such concept we suggest the use of cameras monitoring the environment as well as an underlying system similar to the one described in Trajkovski et al. (2005). Such system will allow agents to explore the environment and incorporate the layout of the confinements as well as objects that may pose a threat or as an obstacle to robots.

Data Flow and Storage

Another important issue that needs to be discussed is the flow of information and the level at which storage for an analysis purpose occurs. Figure 3 shows a diagram of where in the system the database integrates.

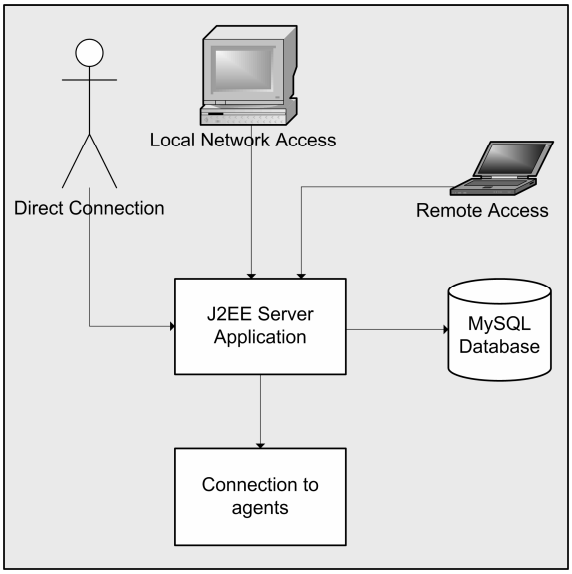

Figure 3: Database connection

The database that we will use is a MySQL database. Such database will store all the information that relates to the programs being run within the server application. Such database will also store information from the agents. We decided not to allow the agents to interact directly with the database for multiple reasons. First of all, it would add a layer of complexity to the structure that controls the robot. Some agents have quite a limited amount of computing power, so it would only take away from the performance of the task to burden the hardware with storage calls, more than with network communications and control management. The second issue that would arise is that there needs to be a method of synchronization among all the agents, thus using the network and hardware resources to carry out redundant tasks. Since the agents will have to transmit the status of various parameters back to the server application that will, in turn, send it to the controllers for performance monitoring, we will also include a data store at that point. This will alleviate the number of tasks every single agent needs to perform, and also the server will control the timestamp associated with each data store strictly. The timestamp is of vital importance for reconstructing simulations for analysis purposes.

Applications in Education and Ongoing Research at CARoL

The first of the many research projects carried out by undergraduate and graduate students involves the use of

computer vision and image processing applied to telerobotics. This branch of robotics deals with the control of robots from remote locations. Such control cannot be achieved without a framework that allows the user to track the robot in its location in order to control and better direct its movement. In order to create such a framework, we use principles of computer vision and image processing to let the remote user see where the robot is located and also what the robot "sees". We will achieve our final goal by means of incorporating three smaller projects. The first involves tracking of the robot within a confined area by means of a simple webcam. The second project will deal with the retrieval of images collected by a camera mounted on a robot. The third project will add the storage of the image within a database and the upload of such images to an applet that the remote user will use in order to control the robot.

Another project worthy of mention involves navigational map learning by context chaining and abstraction. Efficient navigation of one's environment is a fundamental requirement of a successful mobile robot. Ideally an agent's interactions with an unmarked environment should build reliable spatial relationship information without the aid of foreknowledge. Problems arise when different parts of the environment "look" similar to the agent, confusing the agent as to its true position. This problem has become known as 'perceptual aliasing'. In this research project this problem was approached by introducing and investigating the chaining of dynamic virtual landmark identification in the agent's environment. The learning method introduced, called "context chaining" is an extension of the Interactivist-Expectancy Theory of Learning (IETAL). Experiments with software agent simulations showed the success of this approach, measured by the number of steps required to reach the drive satisfier, cumulative memory size, and the number of surprises encountered.

Some of the research also involves cognitive aspects of robotics when applied to a multiagent society. In this project our goal is to reproduce POPSICLE, a cognitive learning study done with humans, except this time with robots, acting as cognitive agents. The original study had participants navigating discrete environments while the participants learning and intercommunication was monitored. To reproduce this we need a platform that is small enough, but with enough functionality to perform the task required. It seems that one platform is not enough; we have broken ours in two sections: high level and low level. The high level functionality, specifically decision making, data logging and wireless networking, we accomplish this with a PalmOS. The low level functionality, locomotion and localization, we perform with the BrainStem, a PIC-based embedded system. Communication between these two systems is via RS-232. Using this combined platform we hope to create robotic-based cognitive agents able to reproduce the original experiment, including intercommunication.

The university setting that we are a part of values practical applications of research concepts. For this reason, the last project that we will mention in this article if the one that has as goal the creation of a fully autonomous tour guide ER1 robot. This robot is intended to be used for giving tours of the cognitive agency and robotics lab (CAROL) and portions of the department of computer information sciences at Towson University. The challenge for such a project is to combine industry like mobile platforms with techniques for mobile robot navigation interactions from contemporary research. The goal of the project is to maximize the autonomy and interactivity of the mobile ER1 platform while insuring higher robustness, reliability and performance. The result is an interactive moving machine with an elaborate Java end that can operate in human environments and interact with humans.

Conclusions

This article introduced a framework that is adaptable enough to allow learning as well as research activities to run. We looked at several aspects ranging from the language of choice to the layers associated with this framework, and we have analyzed how each component will fit. We believe that the approach proposed in this report will be also affordable for low-budget laboratories to start a program in robotics education.

References

Andronache, V., and Scheutz, M. 2005. ADE – An Architecture Development Environment for Virtual and Robotics Agents. *International Journal of Artificial Intelligence Tools*.

Gagnon, E. 2002. A portable research framework for the execution of Java bytecode. Ph.D. diss., School of Computer Science, McGill University.

Greenwald, L., and Artz, D. 2004. Teaching Artificial Intelligence with Low-Cost Robots. *Accessible Hands-on Artificial Intelligence and Robotics Education, AAAI Spring Symposium Technical Report SS-04-01* 35-40.

Hsiu, T., Richards, S., Bhave, A., Perez-Bergquist, A., and Nourbakhsh, I. 2003. Designing a low-cost, expressive educational robot. In *Proceedings of the International Conference on Intelligent Robots and Systems* 3:2404-2409.

Kiniry, J., and Zimmerman, D. 1997. A hands-on look at Java mobile agents. *Internet Computing* 1(4):21-30.

Maxwell, B. A., and Meeden, L. A. 2000. Integrating robotics research with undergraduate education. *Intelligent Systems and Their Applications* 15(6):22-27.

Pomiers, P., and Dupourqué, V. 2006. Modular Distributed Architecture for Robotics Embedded Systems. In *Proceedings of the First National Workshop on Control Architectures of Robots*.

Scheutz, M., and Andronache, V. 2003a. Apoc - a framework for complex agents. In *Proceedings of the AAAI Spring Symposium*. AAAI Press.

Scheutz, M., and Andronache, V. 2003b. Growing agents – an investigation of architectural mechanisms for the specification of "developing" agent architectures. In *Proceedings of the 16th International FLAIRS Conference*. AAAI Press.

Simmons, R., Singh, S., Hershberger, D., Ramos, J., and Smith, T. 2001. First Results in the Coordination of Heterogeneous Robots for Large Scale Assembly. In *Lecture Notes in Control and Information Sciences*, 323-332.

Sun Developer Network 2006. Java Technology. Available at http://java.sun.com.

Trajkovski, G., Schlosburg, J., Whitman, B., and Vincenti, G. 2005. Building Infrastructure for an Honors Research Robotics Lab. In *Proceedings of the Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, 352-357.

Weinberg, J. B., Engel, G. L., Gu, K., Karacal, C. S., Smith, S. R., White, W. W., and Yu, X. W. 2001. A Multidisciplinary Model for Using Robotics in Engineering Education. In *Proceedings of the 2001 American Society for Engineering Education Annual Conference & Exposition*.

White, W.W., Weinberg, J.B., Karacal, C., Engel, G., and Hu, A. 2004. Using Robotics to Teach Integrated System Design via Multidisciplinary Teamwork. *Journal on Educational Resources in Computing*.

Wong, D., Paciorek, N., and Moore, D. 1999. Java-based Mobile Agents. *Communications of the ACM* 42(3)92-ff.