

REACH platform -- Remote Access to Smart Home Facility Based Computer Science Laboratory

Carl K. Chang, Hen-I Yang, Igors Svecs, and Johnny Wong
Department of Computer Science
Iowa State University
Ames, IA, USA
{chang, hiyang, isvecs, wong} @ cs.iastate.edu

Abstract – The purpose of the REACH (REmote ACcess to smart Home facility) platform is to allow students to learn about basic principles of computer science and software engineering practices, and gain hands-on experience through observable effects of the computing systems in a familiar setting (home). It utilizes virtualization to encourage group collaboration and grants anytime, anywhere access to the smart home facility. It incorporates dynamic binding capability that allows students to either use the sensors and actuators hardware they check out and install locally for development, or to conduct experiments in the smart home facility. Once the development is completed, the programs can be deployed remotely, and students can make observations through web cams strategically situated in the smart home facility and the logs of the system's operations and users' activities. The REACH platform is designed to support a large spectrum of computer science courses, from the equivalence of computer science lab 101, interdisciplinary projects, all the way to research projects in artificial intelligence, pervasive computing, and human computer interactions.

Index Terms – Virtualization, Remote lab, Pervasive computing, Smart home, Interdisciplinary courses, Computational thinking

I. INTRODUCTION

Most non-major students often consider computer science to be dull and geeky. Many negative stereotypes surround computer scientists even when computers have become ubiquitous in all aspects of the modern society. To imbue the critical computational thinking (CT) in students of other disciplines, and to counter the misperceptions regarding computer science, we have developed laboratory modules for computer science courses and the corresponding lab facility based on the smart home technology.

For a long time, the instillation of the fundamental CT has been resorted to either taking beginners' programming language courses in Visual Basic or Java, or introductory lectures on computer science. Most of the learning activities both within and outside of the classrooms lean on simple programming exercises, which often only display results on screen, and bear little practical relevance to students' own life experience. In contrary, an introductory lab in computer science would promote students' interest and engagement through hands-on, observable activities, with the effects similar to the introductory labs in physics and chemistry.

This is the reason why we have chosen the smart home facility to be the underlying platform for REACH. It bases on

an environment familiar to all students (i.e., home), generates real and observable effects and benefits, naturally facilitates interdisciplinary collaborations, and has the potential of generating outcomes that can help real people in the real world. This definitely presents an advantage to students who have just started learning more about computer science to be more than just a casual user of computers, and those who begin pursuing cross-disciplinary research or project ideas.

REACH allows students to observe through web cams strategically situated around the smart home facility and the web-based control panel and dashboard. Students will be able to observe with their eyes how the logic in program modules dictates the way sensors collect the data and actuators affect the environment, and learn to influence or alter this behavior via modification of computer programs.

A number of technical improvements have been incorporated into REACH to enable the pedagogical advances and supports new classroom activities.

- Maintains cross-project isolation and separation between development and experimentation environments, which minimizes unwanted interference between independent projects, and provides a shared development environment to facilitate collaboration within teams.
- Provides a comprehensive, individually customizable development environment in a virtual machine image that contains a complete set of development tools, documentation, and collaboration utilities.
- Takes advantage of virtualization to enable seamless migration between development and experimentation modes without requiring changing code by having dynamic binding between services and hardware.

A series of classroom activities have been designed to take advantage of the REACH platform, which embody the following pedagogical innovations.

- Enables a pedagogical approach that encourages a more interactive and freeform exploration for introductory computer science courses.
- Generates observable effects of the software artifact in a familiar setting.
- Aligns the learning activities with the students' career goals and prior accumulated knowledge.

For computer science and engineering majors, one of our primary goals for this project is to develop lab modules that can allow smart home laboratory to be used throughout the undergraduate computer science curriculum, including introduction to CT, programming languages, software

engineering and testing; all the way to conducting senior capstone projects.

We present our work in the following order: representative related works is presented first, followed by a summary of the design considerations and the resulting system architecture. Next, we describe the classroom activities and the associated pedagogical innovations. We conclude this paper by presenting discussions on some interesting design decisions as well as the outlook to future enhancements.

II. RELATED WORKS

A. Virtualization Labs

Virtual Computing Lab (VCL) [1] provides remote access to a dedicated virtual machine with customized software packages. It also provides a reservation system for students. This setting enables students to develop their projects, but its cloud computing nature prevents students from knowing exactly which machine their projects are executing on, which makes sharing hardware devices, such as sensors, and actuators difficult. Additionally, VCL is designed for a single student and does little to encourage the collaboration of students, offering little improvement over the traditional approach where students work at home independently.

B. Remote Labs

Mechatronics Lab [2] features access to live video stream and recorded movies, allowing control using a Java browser plug-in. Step-by-step procedure were given for developing web-based experiments. It allows primary user to manipulate and control devices, and secondary user to observe.

Moon and colleagues proposed a LabVIEW based lab [3] designed to control passive elements in the circuit and observe readings from the equipment. They also presented an integrated view to visualize the design of the circuits with live video feed. Another remote lab project [4] focuses on complementing the real laboratory time with Digital Signal Processing (DSP) experiments using LabVIEW. Both projects serve to supplement specific electrical engineering courses; however, the domain of the labs is not broad enough to promote CT among a wider audience of students. They also focus on using an existing, well established tool suite (LabVIEW) for the very specific subject, while REACH platform can support more generic tools and subjects.

Fujii and Koike created a time-sharing lab [5] using web services and provided integration with TDeLMS (TopDown eLearning Management System) for dynamically course material retrieval. The focus of the lab is CAD-based hardware design and FPGA testing. It is designed as an extension to a regular lab, while the REACH platform can be used for pure online laboratory courses. Similarly to [3] and [4], this lab is limited to a specific domain of hardware design. [5] is only effective in teaching about FPGA hardware, while the REACH platform is more generic, so students are not limited to use specific software or hardware; REACH also focuses more on software, and allows greater changes to be made to the OS (such as installing new drivers or libraries), making the use of virtualized environments more suitable.

MATLAB/SIMULINK based control engineering lab [6] designed by Schmid utilizes JavaScript, Java applets, and virtual reality modeling language (VRML) for animated simulations. This environment does not allow interaction with humans in a real setting, which is a key component in introductory computer science laboratory courses. In comparison, student projects developed using the REACH platform can be easily migrated between development and experimentation modes as needed without any code modification. REACH platform also promotes collaboration while [6] does not.

Fiore and Ratti [7] use a remote laboratory to observe mouse behavior. Students can control the maze, conduct videotaped experiments, and analyze data. They evaluate (1) technical aspects on how to transmit and receive the video recording, and to permit interactive experimental set-up, (2) the practical possibility for the learners to carry out the assigned tasks and to obtain meaningful and consistent measurements, (3) conceptual and technical acquisitions attained by the learners, compared with the levels foreseen by the project, and (4) perceived usefulness, ease of use and user acceptance. This project focuses only on conducting experiments within a specific domain of animal behavior, providing software that has already been developed. It is unsuitable for computer science education, since no programming is involved.

III. DESIGN CONSIDERATIONS

A. Actors

Students (in group and as individuals) and the instructor.

B. Use Cases

1. A student uses the pre-existing tools to complete simple programming assignments.
2. A student uses the observation tools to observe how a program affects the behavior of the actuator, sensor and the smart home as a whole.
3. A student customizes their own workspace to explore non-prepackaged projects.
4. A student deploys their program into the running smart home facility.
5. A student remotely observes the effects of the programs.
6. Students collaborate in group during the development process.
7. Students discuss with the fellow students and instructor the problems and observations during this process.
8. Students maintain a log and a list of bugs and fixes encountered during the development process.
9. The instructor looks into the students' projects to evaluate their progress and provide guidance.
10. The instructor designs and tries out the pre-packaged experiment and programming exercise before handing out to students.

C. Requirements

1. Allows multiple groups to share the same physical smart home facility at different times.

2. Enforces exclusive access to the smart home to only one group of students at any time.
3. Supports students to develop locally, but easily deploy to the actual smart home facility without modifications or reconfiguration of the system.
4. Establishes customizable independent sandbox for each student group to support collaborations, but prevents actions of one group from interfering with another project.
5. Prevents access by any student of one group to the environment owned by other groups.
6. Authorizes access to an environment that includes the necessary middleware, services and bundles to allow the proper operations of the smart home environment.
7. Provides an environment with the appropriate development environment and collaboration tools.

IV. SYSTEM ARCHITECTURE

A. Overview of the Architecture

One of the top priorities of the REACH platform is to support the separation of the preparation and production environments, as shown in Figure 1; in other words, development machines ($SH-DEV\ x$) used by students on a daily basis should be separated from the machine for experimentations and data collection ($SH-EXP$). Each represents a separate smart home environment with its own set of sensors and actuators. Students collaborating on the same project can use remote desktop software to connect to their own smart home environment. When a project is stable enough and students are ready to conduct experiments, their project is migrated seamlessly to the experimentation environment; all servers have the same hardware configuration.

Most pervasive computing systems are inherently distributed, and smart homes are no exception. To provide the complex underlying infrastructure required for development and experimentation, a comprehensive package of the necessary middleware, software libraries and collaboration tools, among others, is by far the simplest solution. In addition, each project may have specific needs for certain resources (such as libraries), and they may be incompatible to those required by other projects. Hence, we adopt OS-level virtualization at the core of the REACH platform.

Our proposed solution is SHADE (Smart Home Advanced Development Environment), a customized virtual machine (VM) image assigned to each group, which consists of all the software components. Each development and experimentation workstation runs a copy of virtualization software, and each group has full control over its own SHADE. This solution prevents interference by groups working on different projects and allows students to make system-wide modifications as necessary.

In addition to $SH-DEV$ and $SH-EXP$, REACH uses an auxiliary server ($SH-AUX$) to provide a variety of utilities essential to its operation. Students can remotely observe their

experiments via web cameras, as well as access sensor logs remotely through a control panel interface.

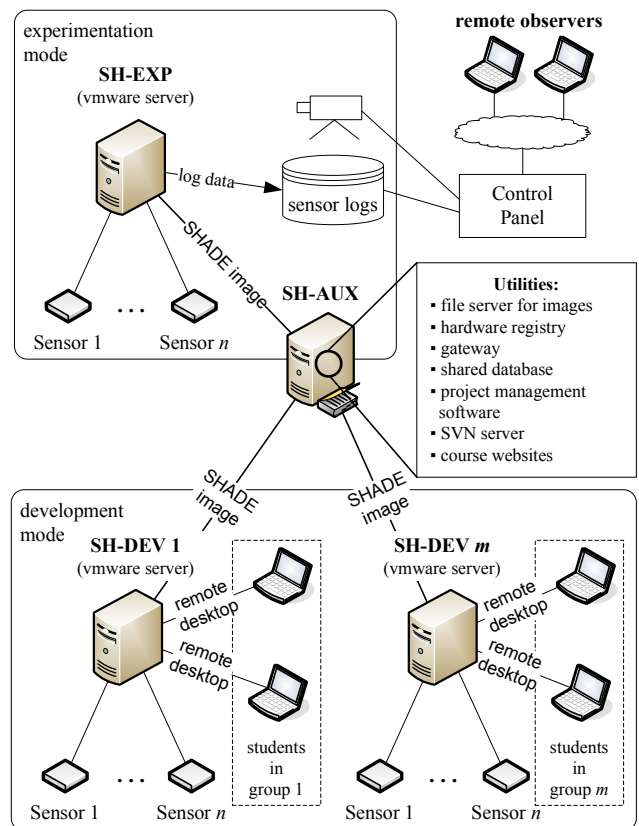


Figure 1. Overall architecture of REACH

B. SHADE Images

Figure 2 depicts the specific software included in each SHADE image.

For the guest operating system, we choose Windows Server 2008 because the REACH platform aims to support multiple concurrent users working on the same system. We opt for Windows Server instead of Linux because most of our students are more familiar with Windows, and it also provides better driver support for a wider selection of sensors and actuators from different vendors. Additionally, some software packages rely on native libraries through Java Native Interface (JNI), which restricts development to the Windows platform.

Fundamental bundles	Database: MySQL	IDE: Eclipse + plug-ins	Version control: SVN
OSGi Framework: Knopflerfish			
Guest OS: Windows Server 2008			
Virtualization: VMware Server 2.0			
Host OS: Windows Server 2008			

Figure 2. SHADE image software stack

Each SHADE image constitutes a comprehensive configurable environment that provides students with all the

necessary tools to work on their projects and conduct experiments. Initially, a master image is created and customized with software relevant to particular courses by instructors. After the master image is tested, it is then replicated and configured for each group of students.

Since the smart home facility is implemented following service-oriented architecture (SOA) paradigm with the OSGi framework at the core of the implementation, the SHADE image includes OSGi framework along with some essential bundles. For instance, hardware driver bundles enable students to program and interact with hardware devices; speech and notification service bundles provide a pathway to communicate with smart home residents; a set of bundles that enables the conversion and communications between web services and OSGi bundles [8] is also included to allow even greater platform independence and flexibility. To enable flexible, easily configurable access to data stores, we create an OSGi bundle as the proxy to databases, configurable to use different data sources. It allows students to use a local database server for development, and switch to the database linked to the Smart Home facility during the experimentation.

Also included in each SHADE image is the bookmarked URL to the Redmine project management web application that provides essential features for collaboration, such as discussion boards, activity tracking, and integration with Subversion version control system. A shortcut to each team's project space on the collaboration website is made available to students. While OSGi bundles and other applications immediately enable students to start development, we also include supporting documentation – tutorials and examples – to serve as a reference, a valuable resource for starting their first projects or as a refresher somewhere down the road.

C. Switching and Binding

Virtualization severs the tight-coupling between the operating system and hardware, offering a mechanism for end users to seamlessly switch between the development and experimentation environments.

When a student group is ready to deploy its project into the smart home facility and conduct experiments, they reserve specific timeslots for exclusive access to smart home online. When the time comes, the SHADE running on one of the SH-DEVS is migrated to the SH-EXP. A system-wide notification service announces system shutdown to all users currently logged in before the migration takes place. Finally, the migrated SHADE discovers and binds to new hardware, so that students' project can continue to function properly, but in the smart home facility instead.

During the initialization process, hardware services indirectly query hardware registry on SH-AUX to obtain an identifier of a device satisfying a set of attributes. A local registry cache service is available on each SHADE to speed up the initial connections. In addition, the caching service can use unique identifier, such as MAC address of a network interface card, to obtain the list of all installed sensors from the hardware registry. The cache is stored on nonvolatile

storage to allow students to run experiments locally without continuous Internet connection.

D. Scalability

Increasing the number of development smart some systems is a simple way to increase the number of student groups that can concurrently work on their projects. The only significant potential bottleneck in our design is the network connection to the SH-AUX hosting VM images and its storage subsystem. One way to mitigate this problem is to introduce load balancing in the lower layers; e.g., utilize link aggregation to increase network capacity and striped disk arrays to increase storage subsystem throughput. Additionally, multiple SH-AUX systems could be organized in a cluster with replication of VM images, with development machines using them in a round-robin way.

Each server can concurrently serve a number of students working on their project. Regular entry-level servers can typically support a group of 2-10 students. Should there be a need for more students to work on the same project concurrently, an upgrade to the server's processor, memory and storage, would be required.

V. CLASSROOM ACTIVITIES

No significant improvements can be made in pedagogy with the advents of laboratory platform and supporting systems alone. The real improvements require a set of carefully designed classroom activities that align with both the objectives of the courses and make best use of the capabilities of the platform. As mentioned earlier, the initial intended audience of the REACH platform includes the introductory computer science courses for students who newly selected computer science as a major, and students who take the first cross-discipline course in computer science. Because of the characteristics of this audience, the activities are designed to be simple but demonstrative to the points, relatable yet thought-provoking. We have designed three categories of classroom activities to be covered in sequence, over the course within a semester.

A. Basics of Computer Programs and Service Computing

The objectives of the first category of classroom activities are as following:

1. To understand the concept of input, output, interface, memory, and service logic;
2. To understand the concept of abstraction, recursion, iteration, and various data structures;
3. To understand service orchestration and composition;
4. To improve reliability through a substitute or backup service;
5. To improve adaptability through service substitution.

The lab activities in this category includes asking students to insert a small piece of code implementing very simple logic (such as loops and branches) to control lamps and TV channels and volume, to modify the interface of the service controlling the microwave, so students can preset the cooking time before going to bed, and to observe the flexibility and resilience when different lamp services with compatible interfaces can be used interchangeably.

B. Smart Environment Service Development

Once students grasp the basic concepts of logical constructs and the service computing, the second category of lab activities aims to demonstrate the usefulness of these concepts and abstractions with the following objectives:

1. To understand how to manipulate the environment through invocation of software services that represent physical sensors and actuators;
2. To design complex composite services using a given set of component services.

We design lab activities asking students to develop an ambient lighting control service that can automatically adjust the lighting based on the luminousness level in the smart home. We provide existing functional luminance sensor service, location service and the various lamp actuation services.

C. Experimentation, Evaluation and Data Analysis

The third category of classroom activities aims to achieve:

1. To learn how to conduct remote observations through web cams and motion sensors;
2. To understand how to create and manage data stores and logs;
3. To construct services for the real-time feedback loops as well as services to extract data for the offline analysis.

Since good evaluation and scientific research begins with good, verifiable hypothesis, we ask students to conjecture a hypothesis, and identify the data needed to support it. We then ask them to design a data collection plan, including specifics of the observation tools such as web cams and sensor logs, and whether the data is to be collected passively or actively. The REACH platform allows students to use a web front-end to specify the criteria, such as the data source, recording frequency, events of interest and handler of these events. The web interface makes it easy to collect relevant data, and the event-driven paradigm allows students to specify the runtime system behavior. Data stored in the log can also be queried and exported for off-line analysis.

VI. PEDAGOGICAL INNOVATIONS

Using REACH to conduct the outlined classroom activities allows us to implement the following innovative pedagogy:

Enables exploratory-based pedagogies in computer science education. Traditionally, CS courses are often taught using direct instruction approach [9], which relies on lectures and carefully designed classroom activities and questions. REACH platform establishes an environment that gives students more freedom to explore, so the instructors can use guided discovery or inquiry approaches [9]. The hands-on, freeform exploration or incubation processes encourage students to be more engaged in the learning of a discipline that many non-major students considered being dull [13].

Generates observable effects of the software artifact in a familiar setting. Typical introductory programming courses rely on textual outputs or some rudimentary displays on screen to demonstrate the effectiveness of the program [13]. For many students unfamiliar with computer science, what is

shown on screen may not be of much interest to them. Many instructors have tried to use tools such as MindStorms [10] or PicoCricket [11] to make the effects of software more observable, while Alice [12] tries to incorporate narration process as a means to teach programming. REACH sets the stage in a setting familiar to all students (home), results in observable effects that are more relevant to their daily lives.

Aligns the learning activities directly with students' career goals and prior accumulated knowledge. Instead of trying to use the generic programming exercises in typical programming textbook, REACH makes it possible for instructors to align the course materials to interdisciplinary students' career goals and prior knowledge. For instance, for an introductory course in gerontechnology, students will design practical computer technology based solutions to enhance the quality of life for older adults, while acquiring necessary CT and basic programming knowledge during the course of the design and implementation. It allows students to pick up crucial computational skills naturally without forcing them to take a separate course in CT or programming.

VII. DISCUSSIONS

A. Alternative Platforms

Several software and hardware configurations were considered to be the REACH platform; each carrying a particular tradeoff. Specifically, we assess different solutions based on how they (1) separate development and experimentation environments; (2) maximize hardware utilization; (3) encourage efficient intra-group collaboration among students; and (4) minimize intergroup interference. It follows from the first two criteria that the process of switching between development and experimentation modes must be as simple as possible.

We consider traditional configuration of using standard local systems accounts for different users on separate computers as a baseline configuration, to be compared to the proposed platform. This option fails to meet goals (3) and (4), as any system-wide change within the scope of one project could potentially affect another project. We also considered using Linux LiveCD distributions that were customized for each type of project to allow students check out sensors and use any available machine in the lab. While this alternative would provide sufficient isolation between groups and maximize hardware utilization, it could lead to software and hardware compatibility issues and impose high administrative overhead associated with the stateless operating system.

Another alternative is to use multiple instances of the OSGi framework on the same computers on a per-group basis, to isolate between groups of students. This option would satisfy all goals of the REACH platform provided that all projects would affect only OSGi framework and associated bundles, as the switching process would only involve launching the appropriate framework. However, this assumption often does not hold when students have to use external software. For instance, almost all projects involve using a database, making pure OSGi framework-level virtualization less attractive.

B. Virtual Machine Image Placement

Having considered potential alternatives for the REACH platform, we concluded that employing operating-system virtualization as described in Section IV is the most efficient way to achieve our goals. Development and experimentation modes could be separated by permanently designating systems as such, and transferring the virtual machine from one machine to another as needed. Hardware utilization is maximized by shutting down idle virtual machines and loading new ones on demand. Collaboration within a group is achieved by sharing a single environment, with changes made by one group member taking immediate effect for others. Finally, intergroup interference is naturally eliminated by keeping virtual machines isolated from each other.

There are several ways how virtualization infrastructure could be organized. First, individual students can work with the VMs on their local computers. While superior when compared to the non-virtualized solution, this approach can dissuade collaboration because of the difficulty in working across several sandboxes. Bundles developed by individual students would need to be manually replicated across workstations, so that other programmers do not run into the risk of using deprecated service interfaces. Additionally, it would lead to underutilization of hardware devices, such as sensors, as each developer would need to check out a complete set, rather than having one set per team.

The second way to organize virtual infrastructure that we considered consists of having a central server concurrently running SHADE virtual machines for all active groups. This approach would simplify management in small-scale deployments, as only one server is needed to support all students. However, an evident disadvantage of complete centralization is inability to scale beyond the needs of a single course. In addition, having all sensors and actuators connected to a single machine would quickly deplete available ports and connectors, requiring administrators to purchase additional expansion cards.

Considering tradeoffs between both methods, we used a hybrid approach that enables intra-group collaboration without suffering bottlenecks of a centralized server.

C. Flexibility to support students in a wide range of classes

The potential of the REACH platform goes far beyond the introductory courses; we also plan to use it for interdisciplinary courses with non-majors, such as gerontology and human computer interface (HCI). It also provides an excellent platform that can be easily ported for studies involving human subjects in areas such as aging in place or gerontechnology. The breadth and complexity of the REACH platform make it feasible to support courses for computer science majors such as programming languages, software engineering, and data mining.

VIII. CONCLUSION AND FUTURE WORK

There is a relatively high barrier, as compared to traditional sciences, in getting new and interdisciplinary students to start to understand and appreciate the critical concepts of computational thinking. To mitigate this problem, we first

created REACH, a platform that can be used to support computer science lab 101, that allows students to learn through hands-on, observable activities in a familiar setting (home); we then designed a series of class activities to allow students to observe the effects of their programs, comprehend abstractions such as services and interfaces, and understand how various components can be put together to provide complex functions in software. An internal testing was conducted with 8 students consisting of both undergraduate and graduate students. The REACH platform is currently being field tested in ComS 486 "Fundamental Concepts in Computer Networking".

Based on the initial feedback, we are currently making minor adjustments and enhancing functionalities to the REACH platform, in preparation for a new interdisciplinary course ComS/Geron 415 "Smart Home Technology for Older Adults" in collaboration with the Gerontology Program starting fall 2010. We plan to collect data from the first offering to evaluate the effectiveness of the platform. Another ongoing effort is to examine how REACH platform can be utilized in an existing introductory course ComS 211 "Conceptual Introduction to Computational Thinking".

REFERENCES

- [1] Sam Averitt, Michael Bugaev, Aaron Peeler, Henry Shaffer, Eric Sills, Sarah Stein, Josh Thompson, Mladen Vouk, Virtual Computing Laboratory (VCL). the International Conference on Virtual Computing Initiative, May 7-8, 2007, pp. 1-16.
- [2] Vikram Kapila, Hands-on Mechatronics Education: A "Brooklyn Poly" Perspective, 2004 Florida Conference on The Recent Advances in Robotics (<http://128.238.129.242/>, <http://mechatronics.poly.edu>)
- [3] Ilhyeon Moon, Saeron Han, Kwansun Choi, Dongsik Kim, Changwan Jeon, Sunheum Lee, Sangyeon Woo, A Remote Laboratory for Electric Circuit using Passive Devices Controlled, ICEE 2008
- [4] F. Barrero, S. Toral and S. Gallardo eDSPLab: remote laboratory for experiments on DSP applications - IEEE Industrial Electronics, IECON 2006.
- [5] Norihiro Fujii and Nobuhiko Koike, A New Time-sharing Remote Laboratory e-Learning System for Hardware Design and Experiment of Digital Circuits, 35th ASEE/IEEE Frontiers in Education Conference 2005.
- [6] Christian Schmid, A Remote Laboratory Using Virtual Reality on the Web, SIMULATION, Vol. 73, No. 1, pp. 13-21 (1999)
- [7] Lorenzo Fiore, Giovannino Ratti, Remote laboratory and animal behaviour: An interactive open field system, Computers and Education, 49 (2007), Elsevier, pp 1299 – 1307.
- [8] José M. R. Álamo, Hen-I Yang, Johnny Wong, Carl K. Chang. Universal Service Composition with Heterogeneous Service-Oriented Architectures, ICOST 2010. To appear in Seoul, Korea, June, 2010.
- [9] Greg Stefanich (eds), Inclusive Science Strategies, Dubuque, IA, Kendall-Hunt Publishing Company, pp. 57, 61 – 73, 2007.
- [10] <http://mindstorms.lego.com>
- [11] Natalie Rusk, Mitchel Resnick, Robbie Berg and Margaret Pezalla-Granlund, New Pathways into Robotics: Strategies for Broadening Participation, Journal of Science Education and Technology 17(1), pp. 59-69, 2007.
- [12] Stephen Cooper, Wanda Dann, Randy Pausch, Teaching Objects-first In Introductory Computer Science, 35(1), SIGCSE Bulletin, 2003.
- [13] Shannon Pollard, Jeffrey Forbes, Hands-on labs without computers, In Proc. 34th SIGCSE technical symposium on Computer science education, pp 296- 300, 2003.