Educational Technologies for Precollege Engineering Education

Mario Riojas, *Member*, *IEEE*, Susan Lysecky, *Member*, *IEEE*, and Jerzy Rozenblit, *Senior Member*, *IEEE*

Abstract—Numerous efforts seek to increase awareness, interest, and participation in scientific and technological fields at the precollege level. Studies have shown these students are at a critical age where exposure to engineering and other related fields such as science, mathematics, and technology greatly impact their career goals. A variety of advanced learning technologies have emerged to enhance learning, promote hands-on experiences, and increase interest in engineering. However, creating and sustaining technology-infused learning environments at the precollege level is a challenging task, as many schools have limited resources and expertise. Moreover, while numerous technology solutions are available to support ambitious engineering-learning goals, choosing the right technology to align to program goals and resources may be a daunting task. In this work, we fill the gap between the applicability of educational implements and suitable teaching methods for precollege engineering. We present an overview of available hardware- and software-based technologies, and characterize these technologies based on criteria such as median price, the type of learning activities fostered, and the required users' expertise levels. In addition, we outline how these technologies align with deductive and inductive teaching methods that emphasize direct-instruction, inquiry-, problem-, and project-based methods, as studies have shown these methods are effective for precollege engineering education.

Index Terms—Educational technologies, learning technologies, engineering education, human-centered computing, interactive environments, robotics.

1 Introduction

There is an increased interest from government organizations, higher education institutions, and citizens' groups to improve existing Science, Technology, Engineering, and Mathematics (STEM) education [3], [40], [103]. Specific to engineering as a profession and as a discipline, these efforts face numerous roadblocks such as the under representation of women and minority groups in the engineering workforce [21], [104], [113], a poor understanding of the engineering profession among younger audiences [21], [104], weak national performance of students in related fields of math and science [106], and the gap between the demand for an engineering workforce and the number of graduating engineers [48], [107].

Although competence in STEM as a whole is widely valued, there is no formal presence of engineering education at the precollege level. The number of precollege students engaged in engineering related experiences is typically reduced to those students lucky enough to have access to summer camps, community outreach programs, or after school programs. In a teachers' survey conducted by the ASEE Engineering K-12 Center, teachers perceived engineering to be less accessible for women, African-Americans, Hispanics, and Native Americans compared to other professions such as law, medicine, and finance [21]. The ASEE report also indicated that teachers believe that majoring in engineering in college is harder than majoring in many other subjects such as english, biology, or social sciences, and this perception is passed from teachers to their students. Due to the lack of formal engineering education in precollege institutions, many students in the United States have expressed no interest in engineering careers and are unaware of the opportunities offered by the engineering profession [4], [21]. Paramount in this effort will be helping students to develop dispositions for engaging in the basic process of scientific inquiry and the application of core engineering concepts [18], [52], [68], as well as overcoming inequities of gender, ethnicity, and socioeconomic background.

A contributing factor to the lack of engineering experiences in precollege institutions is the high attrition rate and a shortage of teachers in related subjects of mathematics and the sciences. Teachers in mathematics and sciences are suitable candidates to take over engineering and other technical courses. However, studies show that math and science teachers in the United States leave their jobs annually at a higher rate (16 percent) than the average teacher-attrition rate (14.3 percent) or the general workforce attrition rate (11 percent) [12]. This shortage further adds to the difficulty of finding qualified instructors to introduce engineering-related courses into the curriculum. A study performed by the National Center for Education Statistics forecasts that by 2015 the United States will face a shortage of 282,720 math and science teachers. The receding cycle between the lack of precollege engineering opportunities and insufficient instructors to teach the discipline contributes to a widening

- *M. Riojas is with the Department of Electrical and Computer Engineering, University of Arizona, 1528 E. 13th St., Tucson, AZ 85719. E-mail: mriojas@email.arizona.edu.*
- *S. Lysecky and J. Rozenblit are with the Department of Electrical and Computer Engineering, University of Arizona, 1230 E. Speedway Blvd., Tucson, AZ 85721. E-mail: {slysecky, jr}@ece.arizona.edu.*

Manuscript received 9 Nov. 2010; revised 10 Feb. 2011; accepted 22 Feb. 2011; published online 31 Mar. 2011.
For information on obtaining reprints of this article, please send e-mail to: lt@computer.org, and reference IEEECS Log Number TLT-2010-11-0124.
Digital Object Identifier no. 10.1109/TLT.2011.16.

gap between the training of students and leaders in the field [48], [104], [113].

There already exist multiple branches of precollege science and mathematics curriculum available worldwide, so is it important to also include engineering? While not all students are interested in following an engineering career, engineering education can foster creativity and innovation in students across a wide range of disciplines, independent of future career paths. In a time where education is leaning toward standardized skills at the global level [17], a country's competitive edge depends on their capacity to be creative and innovative to produce new and unique nonstandard services and products [99]. Simply stated, the success of a country in the global workforce depends not just on the quantity of workers it trains to carry out expected results, but rather in the innovative skills it nurtures in these workers. Engineering education can be a means of fostering the ingenuity demanded by today's market.

It is never too early or late to introduce students to engineering experiences. However, studies reveal that students in middle school[1] are at a critical age where exposure to engineering, or even exposure to a variety of career paths, can greatly impact their future education goals [13], [115]. By introducing students in secondary education to engineering career possibilities, students are able to set up goals and align their middle and high school math and science coursework accordingly, such that they do not face academic deficiencies by the time they enroll in a college engineering program [34]. Three quarters of students who take advanced math in high school previously went through algebra courses during middle school. Students with strong backgrounds in STEM education are more likely to pass advanced placement tests and successfully graduate from higher education institutions [12]. Furthermore, studies have found that female middle school students express more interest in nontraditional fields such as law and engineering [112], whereas this interest fades in high school. One of the reasons for this loss in interest may be the fact that women begin to show less confidence in their mathematical and science abilities as they advance from middle to high school [22], [60] and therefore do not believe majoring in engineering is possible for them [37]. In contrast, studies focusing on women who are engineering majors show that precollege exposures to engineering influenced their decisions and were useful particularly in female students who had no parental encouragement, thus helping them to make informed decision about their future careers [35].

This article is organized to contribute to the discussion of how consumers and developers of educational technology can create conditions and tools to inform, attract, and instill engineering concepts and skills in precollege students. Our view of educational technologies encompasses hardware and software educational implements as well as pedagogical techniques for mediating learning. First, we present a review of engineering concepts commonly taught at the precollege level along with effective teaching strategies derived from the Bloom's Taxonomy [10]. We follow this discussion by an analysis of a representative subset of hardware and software educational implements currently used within the educational domain to further enhance learning, promote hands-on experiences, and increase interest in engineering. We conclude this work with a discussion with a review of how available educational implements can be used to meet theoretical requirements, thereby creating a cohesive learning experience while reaching larger audiences.

1. In the US the typical age of middle school students ranges from 11 to 14 years old.

2 Teaching and Learning Engineering in Precollege Institutions

One of the fundamental questions to address is which engineering concepts are beneficial to teach precollege students who oftentimes lack the educational background needed to engage in many of the curriculum objectives of a college level engineering course? Moreover, given the many different branches of engineering, what engineering concepts are useful for precollege students regardless of the specific engineering discipline (e.g., civil engineering, computer engineering, or chemical engineering) or nonengineering discipline pursued in college? Several works have tackled these questions [18], [52], [68], with the resolution to teach concepts that are useful across various engineering disciplines. Merril et al. [68] define core engineering concepts as COPA concepts representing *constraints*, *optimization,* and *predictive analysis*. *Constraints* are defined as "boundaries for what you can do and the parameters you have to stick to," *optimization* is defined as "the best solution to a problem balancing tradeoffs between competing factors" and *predictive analysis* as "mathematical or scientific principles that are used before the artifact or problem is completed." Similarly, the National Academy of Engineering (NAE) and National Research Council (NRC) have defined core engineering concepts as *systems engineering* and *optimization* of products. *Systems engineering* emphasizes the analysis of structure behavior functions, the study of systems' emergent properties, processes, subsystems, and interactions. Concepts related to *optimization* include, but are not limited to, the analysis of multiple variables, tradeoffs between desirable features, and the consequent side effects of implemented products. In addition to instilling knowledge of core engineering concepts, the development of fundamental skills such as the ability to draw and represent systems and their behavior, perform valid experiments and identify test cases that can accurately prove the fulfillment of product requirements are needed [52]. Furthermore, Custer et al. [18] defined a set of 14 core engineering concepts (design, modeling, constraints, innovation, systems, optimization, experimentation, prototyping, tradeoffs, analysis, problem solving, functionality, visualization, and efficiency) that are coherent with the aforementioned propositions [52], [68]. A commonality between the suggested engineering concepts is their demand for cognitive skills that goes beyond learning facts and recalling. Core engineering concepts also demand high-order cognitive skills such as critical thinking, analysis and problem solving; acquiring these skills is a challenge in precollege institutions where formal engineering curriculums and teaching resources are scarce.

Educators often turn to the Bloom Taxonomy as a benchmark to gauge the development of high-order

TABLE 1
The Bloom Taxonomy-Cognitive Domain

1. Remember
   a. Recognizing
   b. Recalling
2. Understand
   a. Interpreting
   b. Exemplifying
   c. Classifying
   d. Summarizing
   e. Inferring
   f. Comparing
   g. Explaining
3. Apply
   a. Executing
   b. Implementing
4. Analyze
   a. Differentiating
   b. Organizing
   c. Attributing
5. Evaluate
   a. Checking
   b. Critiquing
6. Create
   a. Generating
   b. Planning
   c. Producing

cognitive skills. The Bloom Taxonomy is a conceptual framework that classifies learning goals and objectives in the affective, psychomotor, and cognitive domain, which is commonly used in the development and evaluation of new curricula. The cognitive domain in the Bloom Taxonomy includes six levels of learning activities (Table 1). Each level is associated with a list of verbs that describe specific cognitive functions in the learning process [10]. In Level 1 of the Bloom Taxonomy, Remember, retention of information is achieved when educational tasks are focused on learning and recalling facts. Transition from Level 2 to 6, Understanding to Create, requires the transfer of learning, in which the learners' ability to use knowledge in a different setting from the learning environment. This transition is fostered when educational tasks include experiences that demand deep understanding of a concept and the development of high-order cognitive skills. Both, information retention and transfer of learning are essential for meaningful learning and successful problem solving [64]. Therefore, a full coverage of the Bloom's Taxonomy is expected in any competitive engineering curriculum.

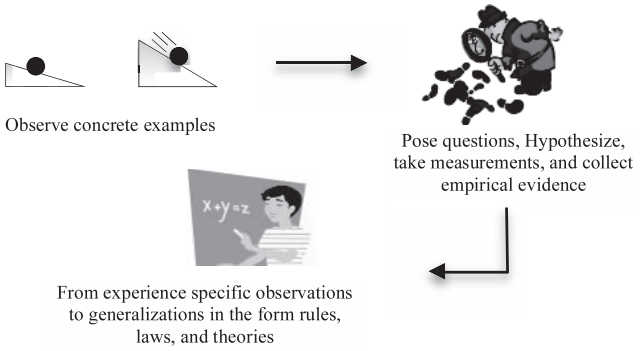

Fig. 1. Inductive teaching methods.

While a multitude of research has been devoted to developing effective teaching methods, we have identified three teaching methods that in ensemble constitute our suggested approach to precollege engineering education. Rather than advocating for one teaching method of instruction, we suggest that a balance between deductive and inductive instruction can better accommodate for a comprehensive approach to the Bloom's Taxonomy. In general, deductive methods rely on highly guided learning activities, visual and oral presentations and elaborated explanations of concepts and the best techniques to learn these concepts. In contrast, inductive teaching methods see learning as a process of knowledge construction and expose students to concrete experiences to foster understanding of abstract concepts. During this inductive process, students have the opportunity to pose questions, hypothesize about current phenomena, take measurements, and collect empirical evidence to proceed from experience-specific observations to generalizations in the form of governing rules, laws, and theories (Fig. 1). Instead of providing or conveying information, the teacher's role in an inductive teaching environment is facilitating and challenging students in order to encourage higher level thinking. Using the definitions of 1) Direct Instruction, 2) Problem-/Inquiry-Based Learning, and 3) Project-Based Learning we identified the driving motivation for each method, central and peripheral activities, and how they relate to the Bloom's Taxonomy (Fig. 2). Moreover, we argue that by having deductive and inductive instruction, respectively, we can address well- and ill-structured aspects of the engineering discipline. The effectiveness of deductive methods to teach well-structured concepts such as facts, laws, principles, and

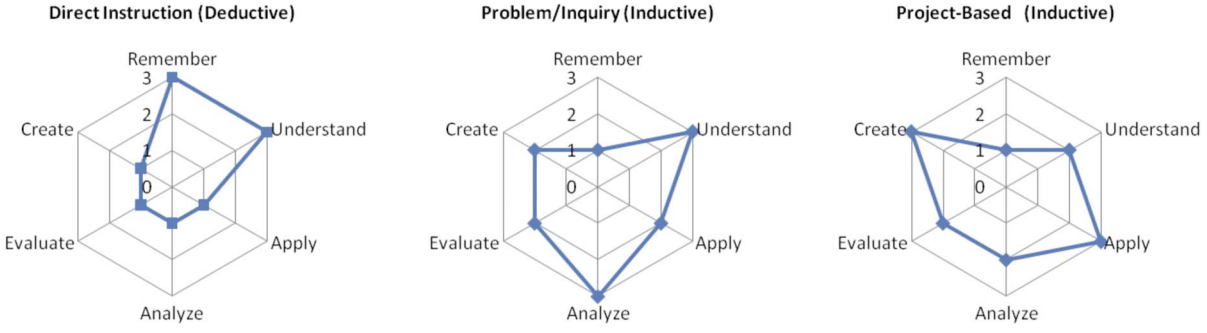

Fig. 2. Bloom's Taxonomy coverage by deductive and inductive teaching methods. A Level 3 in the radar graph represents the methods' driven motivations; Levels 2 and 1 are central and peripheral activities, respectively.

theories has been demonstrated [54]. However, using only deductive methods will not suffice to learn in ill-structured knowledge domains where the conditions for learning cannot be predetermined. Ill-structured concepts are highly influenced by the contextual factors of unconstrained and real-world problems and for which inductive methods are believed to be superior [102]. In the following sections, we define each of these methods as they apply to precollege engineering education.

2.1 Direct Instruction

Direct Instruction [54] is a deductive teaching method where learning is viewed as a function of change in the students' long-term memory. *Direct Instruction* is the most-common teaching method in precollege institutions. Teachers provide students with elaborate presentations that fully explain the concepts of interest. Students then have the opportunity to practice and acquire skills or knowledge under the teacher's supervision in close-ended activities having a predicted outcome. Direct instruction is aimed at acquiring structured, factual, and algorithmic procedural knowledge. Therefore, its application is suitable when students are novices to a discipline and require strong instructional guidance to build a knowledge base that will allow them to effectively work in more autonomous ways. Although considered incompatible with constructivist learning theories and somehow overused, the unique advantages that deductive instruction offers to novice learners make it imperative to include *Direct Instruction* in our recommended approach to precollege engineering education.

2.2 Problem- and Inquiry-Based

Problem- and Inquiry-Based are two inductive teaching methods that share many traits when applied in precollege engineering, often making these methodologies indistinguishable [67], [76], [65], [83]. The underlying motivation of *Problem-* and *Inquiry-Based* methods is the acquisition and analysis of knowledge needed to understand complex concepts, providing learning experiences elicited by questions or problems. Knowledge is constructed through the process of finding a solution to the problem or an answer to a question. The solution to the problem is less important than the knowledge acquired by students through its construction. Oftentimes, the problem description is purposely ill-structured and open-ended. *Problem-* and *Inquiry-Based* methods stipulate no concrete subject matter learning objectives; students are not explicitly required to learn a specific set of facts or formulas. Rather, students perform a self-directed learning cycle where they determine the topics to be learned. First, by analyzing the problem or question at hand, then by identifying the corresponding learning issues that students perceive as relevant to determine a solution. The identification and attention to these learning issues are the essence of the self-directed learning cycle. Students are able to reexamine the problem with a new level of acquired knowledge, repeating this learning cycle whenever new learning issues arise. Because no explicit learning objectives exist, the problems proposed to students have to be designed in such way that they indirectly involve the learning of relevant subject matter concepts and principles. At the precollege level, problems are typically constructed to entail learning topics required in the state and federal education standards, with the identification of learning issues typically regulated by the instructors.

Real-world problems are favored in *Problem-* and *Inquiry-Based* methods, as students are able to analyze these problems from a variety of perspectives without showing inconsistencies. Moreover, as opposed to fictional problems, real-world problems can provide a greater level of ownership and familiarity with students. Ownership of a problem arises when the proposed problem is personally relevant to the learner and not important just because it is a requirement to obtain a good grade. Educators recommend using problem statements that are ill-structured, avoiding inclusion of only key information, which would bias all learners to the same solution. Problem statements should also include information or questions that may not necessarily be relevant to determining a solution [86], [98]. Developing successful *Problem-* and *Inquiry-Based* learning activities can be one of the most challenging inductive methods to implement. Learning is directed by individuals who analyze the problem from their own perspective, as such, there is no guarantee that all the desired topics will be covered by everyone's experiences. While *Problem-* and *Inquiry-Based* methods can pose a number of challenges to educators and students the advantages are often greater, providing authentic experiences and increased interest and motivation [86].

2.3 Project-Based

Project-Based is an inductive teaching method where students are driven by the application of knowledge. The learning activities are motivated by the creation of an end product which is the centerpiece of the curriculum, reflecting real production activities, excluding endeavors that are not directly related with the design and construction of the final products [11], [56], [110]. From the instructor's perspective, the end product represents the students' resulting state of knowledge [11]. Instructors examine the characteristics and behaviors of the students' final designs as well as observe how students improve upon defective products derived from wrong premises. Projects must have a large dynamic range of improvement where participants can clearly observe how the performance of prototypes improves dramatically between design iterations and can determine which designs are superior [97]. Thomas [110] identified five defining features of project-based methods: centrality, a driving question, constructive investigation, autonomy, and realism. The idea of centrality encompasses the in-depth exploration of a particular discipline's fundamental concepts, with the construction of the final product motivating all activities. Unlike *Problem- and Inquiry-Based* methods, intentional exposure to unrelated concepts and facts is not included. Rather, a driving question is posed that focuses the related concepts and principles of a discipline that students must understand. Development and implementation of a final product requires constructive investigation, entailing the acquisition of new knowledge and skills. Students have a high degree of autonomy and are not provided with a predetermined path or expected outcome. Projects involving cookbook style instructions, defining a step-by-step approach leaves little opportunity for students to develop their

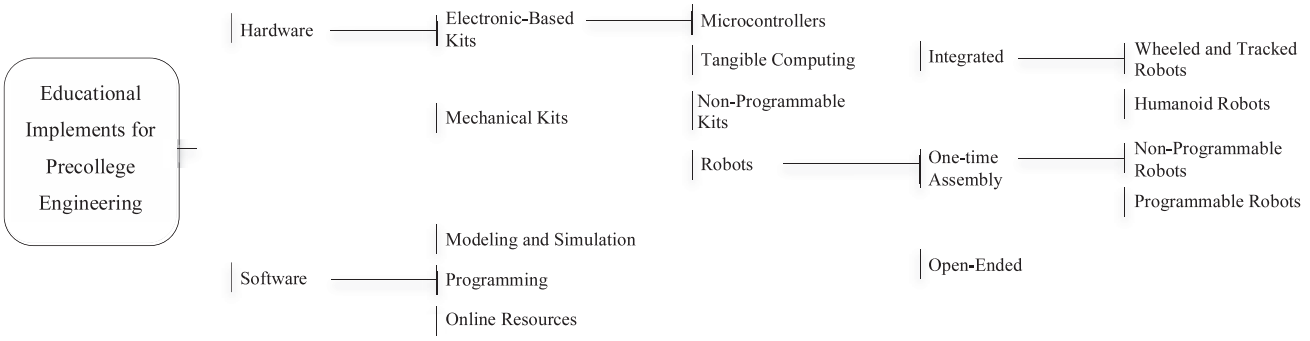

Fig. 3. Taxonomy of educational implements for precollege engineering education considered within this survey.

own solutions are considered an exercise rather than an instance of a project-based module. Finally, project goals should offer a level of authenticity or realism that expose students to the everyday challenges of working professionals, specifically the higher order cognitive skills required to generate new ideas, reflect on experience, and make project decisions.

In summary, meaningful learning can be achieved in precollege engineering education by aligning curriculum goals to target each of the levels outlined in the Bloom Taxonomy, through integration of deductive and inductive teaching methods into precollege engineering curriculum. Based on the literature dictating how to mediate human activity for effective learning in precollege engineering, we additionally present an analysis of promising educational implements that have been specifically developed and/or proved to be effective in teaching engineering-related concepts to novice learners.

3 Educational Implements for Precollege Engineering

In addition to being cognizant of the appropriate teaching methods for engineering education, learning experiences can be further enhanced if supported with complementary educational implements. Suitable educational implements for precollege students will expand their working domain and capabilities, allowing students to successfully manage and accomplish engineering projects by compensating for deficiencies in skills and knowledge typically learned later in high school or college curricula. The support provided by these educational implements should not be considered a replacement for the assistance that can be given by an instructor. Instead, educational implements should be viewed as supportive instruments with the potential to enhance learning by undertaking learners' cognitive challenges, providing access to real-world problems, connecting individuals, and expanding curricula with meaningful and exciting topics for learners [8]. Educational implements coupled with the appropriate pedagogical strategies can address the demands of precollege engineering education. Thus, we classify a subset of educational implements currently available for precollege engineering (Fig. 3) based on the following criteria: 1) hardware versus software platforms, 2) integrated versus one-time-assembly products, and 3) programmable versus nonprogrammable. The following sections elaborate upon these educational implements and their corresponding classifications. In some cases, this basic order is changed to provide more meaningful categories for our discussion.

3.1 Hardware-Based Educational Implements

Hardware-based educational implements dominate the domain of precollege engineering with a wide variety of products that differ in price, afforded learning activities, accommodated expertise levels, and technology requirements. Table 2 provides nine categories of hardware-based products along with the median price, fostered learning activities, and the users' expertise levels they distinctly support. Learning activities are organized in three subject areas: *Electronics*, *Programming*, and *Mechanics*. Furthermore, activities offered by each product are also rated according to their required level of expertise with entries ranging from Beginners (BEG), Intermediate (INT), or Not Available (N/A). Assignment of a N/A rating indicates that the product does not offer activities related to a subject area. In *Electronics*, a BEG classification describes a learner with conceptual knowledge of electronics, while INT indicates a learner possesses procedural knowledge of passive and active electronic components. In *Programming*, BEG requires users have basic knowledge in the development of algorithms, while INT users are proficient in a textual programming language. In *Mechanics*, a BEG mandates users are able to follow step-by-step instructions to build a mechanical device, while INT users can perform open-ended mechanical design activities. Products can offer activities for more than one expertise level. For example a robotics product that allows programming by developing algorithms using a graphical language will be assigned an expertise level of BEG in *Programming*. However, the same product may additionally allow programming via a textual programming language such as Java, thereby also yielding an INT classification.

A cluster analysis[2] of the identified hardware educational implements based on *Price* and number of *Supported Expertise Levels*[3] is shown in (Fig. 4). Samples are grouped into three clusters, LOW, MEDIUM, and HIGH depicted in the graph with "+", "x", and "o" symbols, respectively. It is

2. Six products were excluded from the cluster and correlation analysis. Three products were considered outliers due to their high prices and there was no price available for the others.

3. The *Supported Expertise Levels* was calculated by adding each of the group of learners supported by the product, which ranges from 0 to 6 indicating Beginners and Intermediate learners for *Electronics*, *Programming*, and *Mechanics*.

TABLE 2
Expertise-Level Accommodated by Hardware-Based Educational Implements

	Sample Population	Price	Electronics	Programming	Mechanics
Integrated/Programmable Wheeled and Tracked Robots	12	$150	BEG	INT	N/A
Integrated/Programmable Humanoids Robots	4	$495	BEG	BEG	N/A
One-Time-Assembly/Non-Programmable Robots	4	$38	INT	N/A	BEG
One-Time-Assembly/Programmable Robots	10	$138	INT	INT	BEG
Open-Ended Design/Programmable Robotic Kits	6	$400	BEG, INT	BEG, INT	BEG, INT
Microcontrollers Kits	13	$100	INT	INT	N/A
Tangible Computing	4	$124	N/A	BEG	N/A
Non-Programmable Electronic Kits	10	$84	BEG, INT	N/A	N/A
Open-Ended Design Mechanical Kits	5	$60	N/A	N/A	BEG

Price is in US dollars and is the median for each category. The mode was calculated to determine expertise level. Beginners level is abbreviated as BEG, intermediate level as INT. N/A stands for Not Available indicating that the product does not offer activities in that subject area.

not surprising to see that there is a positive relationship between products' *Price* and the number of *Supported Expertise Levels*. This relationship is better described in a Pearson Correlation analysis that shows a moderate positive correlation of $r = 0.578$, $p \leq 0.01$, $n = 62$ between the two observed variables. However, several products offer similar learning experiences at significantly different prices. In these cases, consideration of additional factors, such as the technology requirements needed to interface to various hardware platforms, are important to inform purchasing decisions.

Table 3 lists the instruction methods supported by hardware-based educational implements. We note that these are not the only methods that can be afforded by each of the listed technologies; the way in which teaching is moderated depends on many environmental factors and is also a function of the instructors' judgment; instead our intent is to highlight the features certain technologies address and how well the requirements of various instruction methods are afforded.

The following sections details several hardware-based educational implements providing one or more illustrative examples for each of the categories found in Table 2, along with references to similar products that an interested reader may consider.

3.1.1 Robotics

The relationship and interest that children have for robots can be compared with the relationships children have with pets and classic toys, such as dolls, dating back millennia,

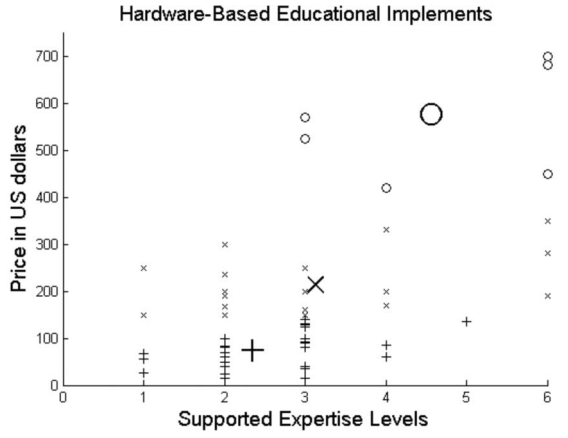

Fig. 4. Cluster analysis of hardware-based educational implements, with "+", "x", and "o" correspond to Low, Medium, and High clusters based on products' price and supported expertise levels. Clusters centers are marked in bigger font size.

TABLE 3
Most Prominent Instruction Methods Facilitated by Hardware-Based Educational Implements

	Direct Instruction	Problem Based	Project Based
Integrated/ Programmable Wheeled and Tracked Robots		X	X
Humanoid Robots		X	
One-Time Assembly/ Non-Programmable Robots	X		
One-Time Assembly Programmable Robots	X	X	X
Open-Ended Design Programmable Robot Kits	X	X	X
Microcontrollers		X	X
Tangible User Interfaces	X	X	
Non-Programmable Electronic Kits	X	X	X
Open-Ended Design Mechanical Kits	X	X	X

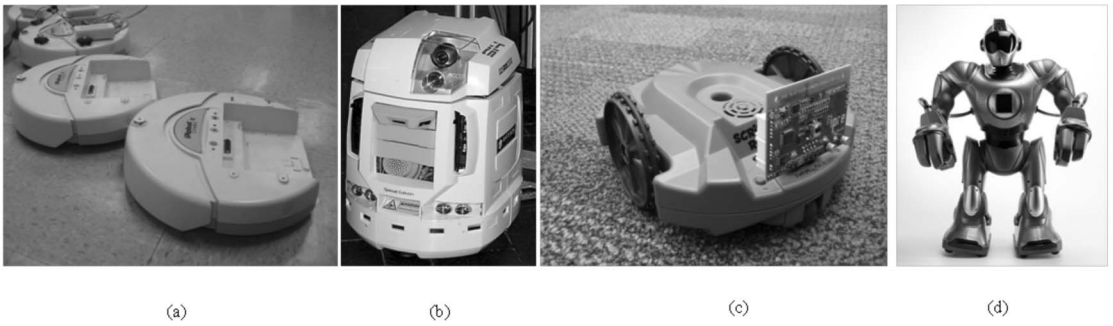

Fig. 5. Sample of integrated programmable robots. (a) iRobot Create. (b) PC-Bot. (c) Scribbler. (d) Humanoid robot: Robosapien RS-Media.

and continuing to have a solid presence in the marketplace [49]. Pedagogically, the interdisciplinary characteristics found in robotics projects can be hard to match in alternative instructional media. Within a single robotics project, students can learn about core engineering concepts as well as science, technology, and mathematics. Furthermore, many robotic kits can be used to facilitate deductive and inductive instruction models. As such, these robotics platforms are expected to become a ubiquitous computing technology over the next few decades [15]. In 2007, the European Robotics Technology Platform estimated that 6.5 million of robots were in operation worldwide, with that number expected to increase to 18 million by 2011 [30]. Miller et al. [71] identified three roles robots can play within an educational setting: robots as a programming project, robots as a learning focus, and robots as a learning collaborator. In the first role, robots are viewed as black boxes that need to be programmed to solve a given problem. Alternatively, when robots are the learning focus, learning topics are oriented toward the design and implementation of robotics such as those found in mechatronics. Finally, when robots take on the role of a learning collaborator, robots include social capabilities to act as students' companions, aides, and intellectual foil [71]. While there are numerous possibilities for robots in education, their cost can be a limiting factor ranging from an initial investment of just tens of dollars to quickly escalating to hundred and even thousands of dollars.

Integrated/Programmable Wheeled and Tracked Robots are ready-to-use robots that embody a chassis connected to a set of wheels or continuous tracks that facilitate their transportation. Ready-to-use robotics platforms provide a complete solution in which the underlying control, sensors, and actuators are already integrated into a package that users can begin to interact with out of the box. These platforms provide a quick and easy method for beginners to program robots as many of the underlying issues are abstracted away. Most wheeled and tracked robots can perform very precise movements and usually can be programmed in more than one way to solve a puzzle-type problem, making them popular in competitive educational projects and tournaments. The activities afforded with these platforms emphasize programming behaviors suitable for a problem-based approach to instruction, typical problems to be solved are programming the robots to follow a trajectory, solving a maze, finding objects or drive while avoiding obstacles. Project-based experiences can also be pursuit by evaluating algorithms and performance using commonly utilized efficiency metrics such as number of coded instructions, time and task success rate.

The *iRobot Create* [46] (Fig. 5a) is an example of an integrated wheeled robot, based on the *Roomba* vacuum design. The *iRobot* is programmed with a personal computer, using a variety of predefined commands to form scripts that can be downloaded to the robot through a serial interface while the robot is tethered or using a wireless module. More experienced users can opt to define complex programs using the C programming language that the robot can read when stored in an attached flash memory. Furthermore, adding peripherals such as a robot arm, light sensors, and range sensors can enhance the iRobot Create platform.

Specifying the desired behavior can be a challenge for many students and teachers, as they might lack the experience in utilizing a general purpose programming language like C, Java, or Basic. To address these difficulties, several robotic platforms provide an alternative graphical programming environment to specify the robot's behavior. For example, the *Scribbler* [82] (Fig. 5c) can be programmed graphically by manipulating icons in *The Scribbler Program Maker GUI*. A number of other wheeled and tracked integrated robots also provide graphical environments to support nonexpert use such as the *Roboni*-i [93], *PC-Bot* [114], and the *Hemisson* [51].

Many of the programmable robots in this category require a host computer in order to control the robot while in use, or to specify the intended behavior and then download the program to the robot. Access to a host computer can be a roadblock to some resource constraint schools or if the intended application requires mobility. The *Snap Circuits RC Rover* [23] and the *Logirobot* [62] are low-cost platforms that can be programmed independently of a personal computer. Behaviors in the *RC Rover* and *Logirobot* are specified using a remote control and can be enhanced with plug-and-play hardware components. The range of activities provided by the *RC Rover* and the *Logirobot* is limited in comparison with other alternatives that admit downloads of more elaborated software programs. Alternatively, *PC-Bot* (Fig. 5b) incorporates a personal computer as part of the platform including a keyboard, monitor, and mouse to specify the desired behavior [114]. Novice learners can easily program the *PC-Bot* using third party applications such as the *Beginners Robot Application Interface and Network (BRAIN)* [94]. *BRAIN* is a windows-based software that allows novices to control robotic operations over the Internet, speech recognition, program movement, and

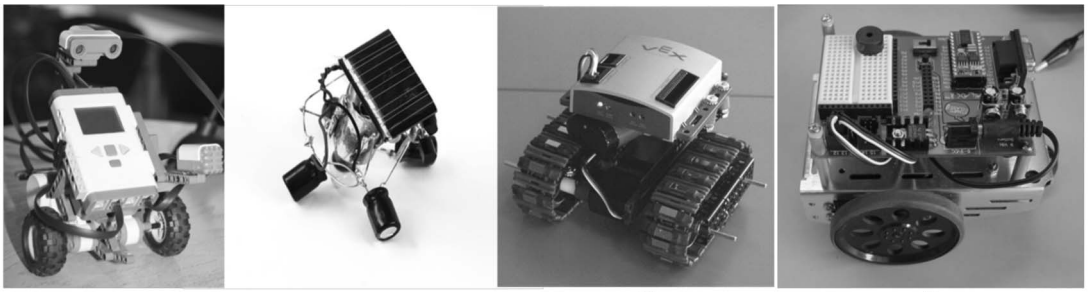

Fig. 6. Sample of component-based robotic kits. (a) Project implemented with Lego Mindstorms. (b) BEAM robot. (c) Project implemented withVEX Robotic Kits. (Picture by Anton Olsen.) (d) Boe-Bot. (Picture by Jeff Avery.)

sensing behavior with a simple scripting language. While *PC-Bot's* processing power and flexibility allows easy adaptation to users of all expertise levels, the tradeoff comes at a high-price point, ranging in the thousands of dollars. Numerous wheeled and tracked robotic platform alternatives exist with similar potential for precollege settings, including the *Amigobot* [73], POLOLU *3PI* [85], *Rovio* [116], *Truckbot* [14], and *SRV-1* robots [105].

Humanoid Robots are a visually attractive type of programmable robots matching closely to the anthropomorphic form that the average person expects from a robot. These robots are characterized by having a defined torso, head, arms, and typically legs (Fig. 5d). While ready-to-use humanoid robots are usually more expensive than the average robotic kit, vendors sometimes offer one-time-assembly versions of their product. Most humanoid robots in the market are differentiated by their available degrees of freedom (one degree per servomotor) as well as how friendly and flexible their programming interfaces are for the novice and intermediate users. Humanoid robots are considerably less precise, less flexible, and weaker than wheeled and tracked robots; in consequence after being programmed to solve a problem, their range of improvement is usually small. Thus, use of these platforms in project-based learning environments is limited, as these projects require evaluating implementations objectively and determining when one design version is better than another. However, because of their programmability, these platforms can solve practical problems involved in problem-based instruction.

Robonova-1 [63] is a humanoid robot system with 16 degrees of freedom available as an integrated robot or as a one-time-assembly construction kit. *Robonova-1* is distributed with a remote control to manipulate the robot along with several preprogrammed movements such as turn, walk, step right, step left, and raise arms. Alternatively, the robot's behavior can be specified using the *RoboBasic* programming environment running in a host computer. Programming behaviors can be specified by rotating dials in the *RoboBasic's* graphical user interface, which represents the positions of the robot's servomotors or by entering code in BASIC programming language. *Robonava-1* can be programmed through several methods, and the option to acquire the platform as an integrated ready-to-use or one-time assembly robot makes this platform accessible to audiences with a high diversity of expertise levels.

While some of the characteristics of humanoid robots make these platforms less common in educational projects and tournaments, many hobbyists and educators find these platforms appealing. *NAO* [2], *iSobot* [45], *Robophilo* [92], and the *RS-Media Robosapien* (Fig. 5d) [117] are other examples of other humanoid robots.

One-Time-Assembly/Nonprogrammable Robots are educational kits that allow users to build robots from the ground up by following a rigid prespecified design. The main advantage of these platforms is their low cost, and as these robots are not programmable, users do not need a personal computer to effectively work with them. Although these platforms are often perceived as nonconducive to producing original solutions and therefore ill-suited for inductive teaching methods, they can be used in direct instruction activities providing hands-on experiences to learn low-level details of dynamics and electromechanical components in a well-structured environment.

The most common examples of one-time-assembly/nonprogrammable robots belong to the *BEAM* bots family (Fig. 6b). *BEAM* stands for biology, electronics, aesthetics and mechanics and is a term that applies to reactive autonomous mobile robots that frequently mimic insect and animal behaviors. BEAM bots are composed of low-cost electronic components, therefore their acquisition cost is less expensive than most robotic alternatives. However, components usually need to be soldered together; and once the final product is built, it is difficult to take apart to reuse components in later projects. Examples of commercially available *BEAM* bots construction kits are *Herbie the Mousebot* [101], an agile wheeled light seeking robot with sensors in its whiskers and tail that allows the Mousebot to back off when it bumps into obstacles, and the *Photopopper* [101] which is a two leg solar powered robot with optical and touch sensors capable of avoiding obstacles. Other robots in this category are the *Weasel* [78], which does not require soldering, and the four-legged robot *Moon Walker II* [77].

One-Time-Assembly Programmable Robots offer the unique advantage of programmability while still providing users with the hands-on experience of building a robot from the ground up. Like other *One-Time-Assembly Robots* their structured designs accommodate activities that can be moderated by direct instruction, requiring the ability to read schematics, soldering, and installing firmware. Once assembled these products offer similar learning experiences as the integrated ready-to-use products with applications in

problem- and project-based teaching. While the average price is oftentimes higher than their nonprogrammable counterparts, these platforms remain less expensive than the integrated ready-to-use solutions.

Sumo robots are a popular choice in the one-time-assembly/programmable robot category. These robots compete with one another to detect and push an opponent robot outside of a ring. The *Sumobot* [81], [82] is a construction kit-based around a BASIC Stamp 2 microcontroller that can be assembled without soldering components together. The *Sumobot* manual explains graphically, and in text, the assembly procedure and additionally introduces topics such as locomotion under program control, sensor theory, the *Sumobot* mechanical and electrical system, and programming concepts using the *PBASIC* language. The *Sumobot* manual also includes a series of suggested programming exercises that help learners to hone their programming skills before tackling a complete solution. Like with most robots in the sumo group, users cannot easily perform significant mechanical alterations to the *Sumobot*. However, its flexibility lies in the powerful algorithms that can be programmed to control the robot as well as the type of sensors that can be incorporated to detect the sumo ring borders and opponents. Other available sumo robotic platforms are the *MarkIII* [50], *Sam R1*, and the *Sumovore* [101].

Equally prevalent in this category are general-purpose robots like the *TruckBot* [14], which can be assembled from a cardboard kit and then coupled with an Android cell phone acting as the controller. Also available are the *BoeBot* [80] (Fig. 6d) which like the *Sumobot* it is based on the Basic Stamp 2 microcontroller and it is distributed with comparable didactical resources, and the *BugBrain* [119] a variation of BEAM bot with programmable features.

Open-Ended Programmable Robotic Kits are probably the most popular education robot platforms today and usually consist of a microcontroller, a software development environment to program the platform, a set of servomotors, and a set of sensors to form a variety of designs. Besides facilitating the engagement in open-ended design demanded in problem- and project-based teaching methods, these products are accompanied with resources to implement suggested projects that can be followed using direct instruction. Although these platforms are typically more expensive than the average robotic kit, *Open-Ended Programmable Robotic Kits* are attractive as they allow the development of original designs employing reusable pieces to pursuit a variety of projects.

The *LEGO Mindstorms NXT* [59] is one of the most popular examples of a robotic kit that enables open-ended programmable projects (Fig. 6a). The main component of the Mindstorm development kit is the *NXT* programmable brick that interacts with a variety of sensors and actuators. Users can program the *NXT* directly using the hierarchical menu displayed on the LCD or by using the *LEGO Mindstorm Education NXT* software on a host computer. In both cases, programming is intuitive as a graphical programming methodology is utilized. Programming can also be done through third party software that will allow text-based code entry using programming languages such as C. This platform not only supports wheeled and tracked robotic applications, but also can be utilized for humanoid applications as well as numerous other electronic systems.

The *Bioloid* [111] robot kit is another comparable open-ended programmable robotics kit and is composed of dinamixels, a central controller, and frames. Dynamixels are actuators that serve as joints within the robot structure and provide points of movement or sensing. Frame components are small plastic pieces used to make connections between the dinamixels, as well as connections to the main controller, and form the body of a robot. The *Bioloid* platform supports three programming packages to accommodate users with different levels of programming expertise. The *Behavior Control Program* allows users to specify the robot's program by dragging and dropping icons into a graphical window, including the use of control flow structures such as if-then-else, jump, and return statements. Alternatively, the *Motion Editor Program* allows users to record the positions of a robot as images. A series of robot positions can then be grouped together to form a program that executes each of the positions in a sequential order. Finally, the *Robot Terminal Program* allows the greatest flexibility in programming behaviors through textual instructions targeted for the more experienced users.

A number of robotic design and development kits are available and include the *Vex Robotics Design System* [43] which offers a classroom lab kit with construction materials for wheeled and tracked robot packages and an extensive pedagogical curriculum (Fig. 6c). The development of wheeled robots is the focus of the *RoboTX Training Lab* [32] and the *RDS- X01 RoboDesigner* [95] kits. Other nonwheeled robots well suited for open-ended projects is the *Robobuilder-Creator 5710k* [91], which provides a flexible robotic platform to build humanoids and animal like robots.

3.1.2 Microcontrollers

The wide use of microcontrollers in education varies from robotic applications and data logger systems to artistic creations. Microcontroller kits provide users with both a reprogrammable integrated circuit with memory and a processor capable of interfacing with different types of inputs and outputs. Microcontrollers are the core of many commercial programmable robots and, therefore are typically included within the platform. Alternatively, when educators do not want to commit to a specific robotic platform or to robotics at all, microcontrollers can be acquired at a lower. Microcontrollers kits provide "intelligence" to mechanical systems, but also can be used as a standalone programming tools. It is possible to use microcontrollers in direct instruction by following step-by-step close-ended projects. However, the processing power of microcontrollers makes these platforms better suited to support original designs in problem- and project-based learning.

The *GoGo Board* [72] specifically addresses the issue of cost by developing a general-purpose microcontroller platform with an open source hardware design freely available to anyone interested in building their own microcontroller kit. The *GoGo Board* design utilizes components that are available in common electronic stores. Additionally, the size of components used within the *GoGo Board* require no specialized soldering equipment, making it possible for the

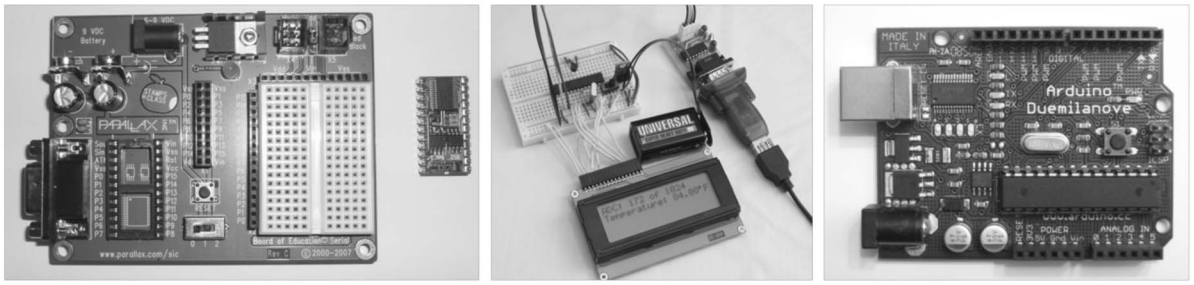

Fig. 7. Sample microcontroller educational kits such as the (a) Basic Stamp II and Board of Education, (b) Nerdkits microcontroller, and (c) Arduino microcontroller.

average person to build their own platform. The price of constructing a board is less than US$50. In addition to making the components accessible to a larger audience with an open source, the *GoGo Board* also supports users of all levels with a variety of programming languages including logo-based programming languages and Active-X compatible languages. While many of the roadblocks associated with integrating microcontrollers into an education setting are addressed by the *GoGo Board* platform, a minor obstacle that may be faced by some users is the need for a Peripheral Interface Controller (PIC) programmer. The PIC programmer is needed as a onetime setup to install the board's firmware. Once the firmware has been installed, the board can be reprogrammed as needed, downloading programs from a computer connected to the board through a serial port. Other available open source microcontroller platforms are the *Arduino* (Fig. 7c) [6] and the *MIT Handyboard* [39].

The *PICAXE* is another low-cost alternative in the microcontroller domain [90]. *PICAXE* is a standard PIC microcontroller with preprogrammed bootstrap code that enables reprogramming of the microcontroller without the need for special hardware. Users are able to program the microcontroller as many times as needed, downloading programs from a computer through a serial port. Programs should be written in the *PICAXE Programming Editor* in either *BASIC* language or as graphical flow charts. The *PICAXE* website contains instructional files to take over a series of suggested projects such as an electronic dice project, a Simon says game and a cyber pet project. The simplicity of the *PICAXE* microcontroller platforms allows low cost prices starting at US$ 20 per basic kit.

Although many projects commonly associate microcontrollers within the domain of robotic projects, microcontrollers can be utilized in other application domains. For example, the *PicoCricket* [108] is a microcontroller platform specifically designed to exploit creativity through interactive artistic projects. The *PicoCricket* kit is composed of the *PicoBlocks* programming software, along with sensors and actuators that allow users to tackle diverse projects combining engineering and art. Sample projects include a stuffed animal that makes a sound when someone pets it or a reaction game that calculates the time that it takes to a player to react after hearing a chirp. These projects can be easily implemented using the *PicoCricket* in part due to the *PicoBlocks* programming software. The *PicoBlocks* development environment is based on a visual language where programs are composed using a set of graphical programming blocks representing sensor inputs and outputs, constant numbers, actions, and control of flow. A significant drawback of the *PicoCricket* is its price (US$250 per kit), which rapidly escalates when considering the number kits required for a regular sized classroom.

The C programming language is one of the most popular languages used to specify the functionality implemented by microcontrollers. C is a general purpose programming language with a great number of compilers and debuggers widely available. However, a few microcontroller beginner kits allow users to work with C as it is often perceived as a more challenging language than BASIC or other visual languages because C has a large number of syntactic elements, sometimes without an obvious meaning [53]. The *Nerdkits* [75] microcontroller platform (Fig. 7b) is an option for users interested in programming microcontrollers using C. The *Nerdkits* includes an Atmel microcontroller, a solderless bread board, an LCD display, along with other electrical components and an instruction book describing sample projects. The *Nerdkits* use an open source software development tool that has not been adapted to beginner users. To compensate the challenge faced by novice users working with a general-purpose programming language and programming environment, the *Nerdkits* website provides a collection of text and video tutorials explaining programming practices as well as fundamental electrical and computer concepts.

While most microcontroller kits require a personal computer to be programmed, a different programming method is available through the *Chip Factory* [89] kit where users can program commercial microcontrollers using a physical interface. The *Chip Factory* allows users to enter BASIC style commands through a on-board keypad and LCD display. The type of programs that can be developed through this interface may be limited, however the option to program a microcontroller using simple commands without a computer may be very attractive to entry-level users.

Many alternatives in the microcontrollers category are available, including the *Phidget Interface Kit* [38] that supports various programming languages and operative systems, and can be used with costumed "plug-and-play" sensors and actuators. The *Basic Stamp II* [79] microcontrollers (Fig. 7a) are designed to operate *Parallax* educational robots such as the *Sumbot* and *Boebot* described in previous sections. The *Snap-Micro* [24] kit offers an environment

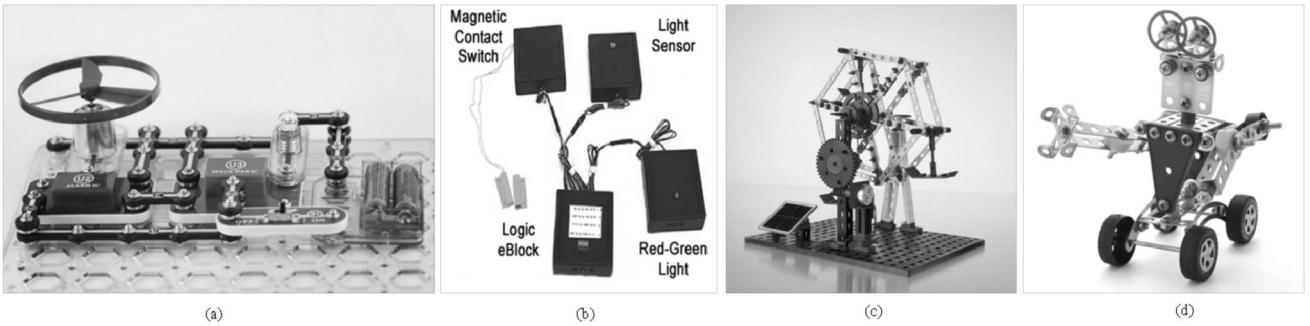

Fig. 8. Sample of nonprogrammable educational kits. (a) Snap Circuits kit. (b) Alarm system implemented with eBlocks. (c) Solar powered Ferris Wheel by Fischertechnik. (d) Model constructed with Meccano.

suited for beginners by allowing flow chart style programming and using *Snap-Circuits* (Fig. 8a) components. The *Picoboard* [109] enables physical interactions with computer animation programs developed in the *Scratch* [100] programming language. The large number of microcontroller platforms available today can accommodate a variety of project needs ranging from application flexibility, cost, and programming environments.

3.1.3 Tangible Computing

Tangible computing is a growing research area that strives to provide users with a naturalistic alternative to interact with computers [44]. Tangible User Interfaces (TUI) uses the physical properties of objects to represent digital information. The users' physical interactions with objects enhanced with tangible interfaces results in the manipulation of the digital information the objects represent. TUIs take advantage of humans' natural ability to grasp sense and manipulate objects with their hands. Tangible programming languages are a common application of TUIs in education. Similar to visual languages, tangible programming languages provide users with a mechanism to express algorithms without ever having to learn complex language syntax. Coding with tangible programming languages is an interactive activity where an algorithm is perceived as a physical shape that can be built by multiple users, allowing collaborative learning. TUIs are easy-to-use tools that can reach many novice learners with no previous technical background, making it feasible for users to follow predetermined steps to produce a project using a direct instruction approach. Problem-based methods can also be implemented however, developing project-based activities might be a more challenging as these interfaces provide a few evident ways to objectively determine when designs are more effective than others.

The *Tern* [42] tangible programming language was developed to enable users to program offline robotic platforms such as the LEGO *Mindstorm* and the *iRobot Create*. *Tern* is composed of a set of wooden blocks, each with a spotcode that represent operators, constant numbers, and control-of-flow structures. These wooden pieces snap together to form models that represent a computational program. *Tern* has no embedded electronics therefore, the compilation process requires taking a digital picture of the program and downloading it to a host computer. Compilation is performed in the host computer by analyzing the spotcodes on each of the pieces that composed the program. Although still dependent on a digital camera and a personal computer the main advantage of tangible programming languages like *Tern* is that they provide an accessible workspace to develop programs concurrently by multiple users.

In addition to tangible platforms that are designed exclusively to replace the need for traditional programming languages, other examples of TUIs can also be used in educational settings. *Siftables* [69] are a platform that integrates wireless sensor technology with a tangible user interface. An illustration of how tiles are used is shown in Fig. 9. The *Siftables* platform is composed of compact interactive tiles with graphical displays that are equipped with accelerometers to sense motion such as being lifted, tilted, and shaken. These motion sensors allow users to interact with the tiles and to manipulate digital information through movements. For example, a user can enter a "yes" input into a siftable tile by shaking it up and down. Alternatively, a user can enter a "no" input by shaking the tile sideways. Each siftable contains IrDA transceivers to detect and communicate with neighboring siftable within a distance range of 1 cm. Longer range communication with other siftables and computers is possible through a wireless radio. The *Siftables'* display allows users to observe in real time any changes performed to the information they represent. Many applications can be designed to run on the platform, including programming languages where "computational results" are visualized immediately after arranging some tiles and also modeling languages with simulation capabilities.

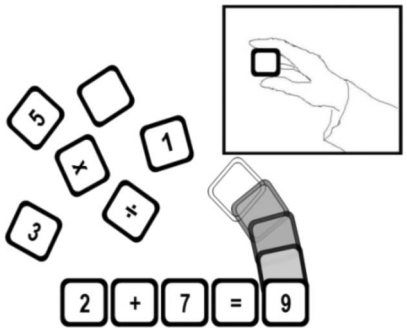

Fig. 9. In Siftables' users can see the results of a mathematical operation on real time by sliding one blank tile next to the equal sign.

3.1.4 Nonprogrammable Electronic Kits

Nonprogrammable Electronic Kits are low-cost products usually composed of a set of plug-and-play modules that are used as building blocks to develop more complex systems. Having a personal computer is not a requirement to effectively use *Nonprogrammable Electronic Kits*. The learning experiences that can be afforded with these products include deductive and inductive methods. Traditionally these kits illustrate basic electronic concepts, however an increasing number of *Nonprogrammable Kits* are also focusing on illustrating sustainable energy concepts.

Learning to design electronic circuits can be overwhelming for novice learners requiring low-level procedural knowledge of how components work and interface with one another. The *Snap Circuits* Kits provide a working platform for the design of electronic systems (Fig. 8a) [25]. The *Snap Circuit* Kits are composed of plastic pieces with embedded passive and active electronics that are color coded and graphically display the corresponding standardized symbols. Users can implement entry-level electronic circuits by snapping plastic parts without soldering or wiring in a breadboard.

The *eBlocks* educational kits, shown in Fig. 8b are composed of a set of fixed function blocks that enable nonexpert construction of embedded system applications [84]. The *eBlocks* provide a hardware wrapper that encapsulates "dumb" sensors or actuator and a microcontroller, which adds computer intelligence, hiding low-level interfacing details and communication protocols from users. Users are able to build customized embedded applications in minutes without extensive programming or electronic training. The *eBlocks* snap together and the order in which blocks are connected specifies the system's functionality. The *eBlocks* domain is composed of sensor blocks capable of monitoring the environment, computational blocks that can perform logic, integer and state functions, actuator blocks that can provide stimuli and communication blocks that provide wireless point-to-point connectivity. The *eBlocks* kits contain approximately 20 different blocks that cover a wide range of projects from a simple doorbell system to a wireless monitor-control network that can detect leaking appliances in a basement and call a user-specified phone number. While the fixed function blocks enable users to quickly build a variety of embedded systems without need to program the underlying microcontroller, the flexibility to create new functions is limited. *Logiblocs* [61] and the *Electronic Blocks* [118] are also examples of nonprogrammable electronic kits. *Logiblocs* are used by a number of primary schools across the United Kingdom. *Logiblocs* offers predefined kits that require following specific designs such the spy kit and the room alarm kit, and general-purpose kits that allow for the development of open-ended designs. *Electronic Blocks* [118] are fixed function blocks that users stack together to produce the desired output. *Electronic Blocks* are aimed at students between the ages of 3 to 8 years old and are limited in use for older students due to the simplicity of their programmed functions. Many of the characteristics of the *eBlocks, Logiblocs,* and *Electronic Blocks* align with the goals found in tangible computing platforms.

The interest for sustainable energy sources is globally increasing and with it the development of educational implements to illustrate related concepts. The LEGO *eLAB Renewable Energy Set* [57] provides lesson plans and LEGO components to build systems that show how energy can be developed from solar, wind, and water sources. Similarly, sustainable energy principles can also be taught by using the *Renewable Energy Kit* [41] which is composed of a turbine module, solar panel, electrolyze, a PEM fuel cell and hydrogen storage system. Many sustainable energy concepts can be exemplified with the *Renewable Energy Kit.* However, these modules have very rigid designs, which make it hard to be used in the development of open-ended projects. Also available are the *Thames and Kosmos Alternative Energy Series* [108] that offers several kits including a power house project, hydropower, wind power, and fuel cell kits. The *Green Science* Series [1] are low-cost kits bound to a specific project such as a dynamo torch or an enviro-battery. The Eco Tech (Fig. 8c) and the Hydrocell Kit, by Fischertechnik, are suitable for close- and open-ended designs, illustrating sustainable energy concepts through model construction [32].

3.1.5 Open-Ended Design Mechanical Kits

As with the Open-Ended Programmable Robot Kits described in previous sections, Open-Ended Mechanical Kits similarly allow the design of original and innovative solutions. Mechanical kits are composed of nonelectronic pieces that are used as building blocks to develop mechanical systems. Open-Ended Mechanical Kits make engineering experiences accessible to a broad audience because they require no previous technical experience in programming or electronics, nor do they depend on computer technology. This advantage is also a limitation as Open-Ended Mechanical Kits permit a narrow range of learning activities usually constrained to the design of applications of simple and motorized machines and static structures. Nevertheless, they can be effectively used to implement deductive learning by performing close-ended design activities and inductive teaching methods when requiring original designs.

The *LEGO Simple and Motorized Mechanisms Set* [58] can be used to study motion and forces by developing mechanical systems with conventional LEGO pieces along with other customized parts such as gears, axles, and motors. The range of activities that are supported can be further expanded with the *Mechanisms Pneumatic Add-on Set,* which includes pneumatic cylinders, pumps, hoses, and valves. Similarly, Fischertechnik [32] provides several kits that also illustrate mechanical and structural engineering concepts including the *Mechanics and Statics* kit, which can be used to design several gear systems and static structures. The *Pneumatics II* kit is akin to its LEGO counterpart. A more specialized, but open-ended product is the *Totally Trebuchet* [33] kit, which allows users to design and build trebuchet models with metallic and wooden pieces that can be assembled using a screwdriver. Other kits for the construction of mechanical models are *Meccano Super Construction Set* [66] used to develop motorized and static designs (Fig. 8d) and the *Engino Mechanical Science* [29] series, which features models of several simple machines.

TABLE 4
Most Prominent Instruction Methods Facilitated by Software-Based Educational Implements

	Direct Instruct	Problem Based	Project Based
Modeling and Simulation		X	X
Programming Learning Environments		X	
Online Resources	X		

3.2 Software-Based Educational Implements

Software-based implements are a popular choice for precollege engineering education and include design, modeling and simulation tools, introductory programming environments, and online engineering information resources. Software tools offer unique opportunities to institutions, instructors, and students. In many cases educational engineering software can be obtained at reduced prices or free of charge. Numerous applications have been designed to teach concepts that would traditionally depend on purchasing expensive or inaccessible hardware if implemented in nonsoftware environments. Instructors can effectively manage classrooms while students work at their own pace on a designated computer, while taking advantage of the embedded aides offered by many software tools. The effectiveness of software-based tools to teach engineering concepts is comparable to the effectiveness of teaching with hardware alternatives [55]. Similarly the creativity, usefulness and originality of end products developed in a software-based environment can also matched those designed in physical environments [70]. As with hardware-based educational implements, we have established a relationship between the different categories of software implements considered and the instruction methods that they most prominently facilitate (Table 4). The technical expertise required by novice users to interact with software tools is usually modest in comparison to many of the hardware alternatives. New users are able to independently learn how to use software implements by following the tutorials distributed along with the tools. Additionally, during usage monitoring aides are also provided within the software environments to guide users. The accessibility of these software implements, however, should not be confused with simplicity, many software tools provide not only opportunities to engage in a wide range of learning experiences, but also the functionality to continually refine and elaborate artifacts in depth.

3.2.1 Modeling and Simulation

Real-world tasks, such as the development of physical prototypes for a user-defined application, can greatly contribute to increased participation and interest of students in engineering courses. However, the time, money, and expertise required to build physical models or prototypes are not always accessible. Thus, an alternative is to utilize modeling and simulation software to engage students by creating abstract representations of systems within the software framework as well as observe the behavior of the proposed system under various conditions. The time and cost invested to develop a series of prototypes can be kept to a minimum, while conceptual ideas such as the benefits and drawbacks of design changes are easily observed. Furthermore, many simulation environments enable students to easily take appropriate responsive measures and observe the impact of this decision in real time.

The learning advantages that modeling and simulation tools facilitate make these software implements suitable for problem- and project-based instruction. Learners can readily design models to solve a variety of problem scenarios while utilizing simulation to evaluate and validate the proposed solutions. Within the domain of precollege engineering education, system dynamics, mechanical, and electrical and computer engineering are common themes found in modeling and simulation software packages.

Systems dynamics platforms provide students with the resources to observe how complex systems change through time, engaging students through modeling and simulation of known structures and rules. Several products are available to represent mental models as system dynamics simulations even when the underlying math describing the models is not well known. For example, *Vmodel* [31] is a qualitative modeling environment targeted toward middle school students. A subset of qualitative representations and reasoning techniques are provided to allow students to represent a variety of processes. For example, students can model and simulate the process of heating water in a microwave or the behavior of an internal combustion engine. The *Vmodel* ontology is composed of a set of entities such as processes, things, substances, parameters, and rates. Additionally, a set of relationships such as contains, does, requires, greater than, less than, increases, and decreases can be specified in each model. Users build models by dragging and connecting entities and relationships in a graphical user interface. The *Vmodel* software platform strives to expose students to introductory modeling concepts and tools, facilitating a smooth transition to more complex mathematical modeling at the high school or college level. *STELLA* (Fig. 10a) [47] is another platform which similarly provides a graphical drag and drop paradigm to create models. Advanced users can additionally access the underlying mathematical representations of the models developed, which are automatically generated by the software platfrom from the specified graphical representation.

Mechanical design software platforms enable students to graphically represent a physical system, or aspects of a physical system, previous to its implementation. The

Fig. 10. Modeling software examples. (a) A system dynamics model in *STELLA*. (b) Shows the design of a circuit by specifying truth table values in *Boole-Deusto*. (c) Nelson Coconut chair from SmartFurniture developed in Google Sketchup.

models can be subjected to visual analysis or observed under different simulated conditions. Similarly, the robustness of these solutions can vary from coarse interfaces targets toward novice users, to industry standard tool suites. *Google Sketchup* [36] is a free software tool targeted toward novice learners, which has an intuitive interface where users can quickly begin modeling 3D objects by drawing object perimeters in a 2D space. With the mouse, users push and pull the edges of their 2D designs to convert these designs into 3D objects. In contrast, high-priced modeling software such as *Autocad* [7] and *SolidWorks* [19] have a greater learning curve, but offer extensive functionality and simulation options that more experienced users would find attractive.

The design of *Electrical and Computer Systems* is a popular topic in precollege engineering education with a variety of software packages available, allowing students to study these systems at different levels of granularity. Electronic systems can be modeled at the gate level, derived from Boolean logic operations and Finite State Machines (FSMs), using tools such as the *Boole-Deusto* [120] educational software (Fig. 10b). Electronic systems can also be modeled using libraries with building blocks preprogrammed to emulate the execution of powerful operations. For example, in *LabView* [74] users have access to predefined modules to read sensor measurements, analyze data, generate DC power supplies, and arbitrary waveforms. *Labview* enables users to design and simulate electronic systems by combining graphical and text programming. While many of the software implements provide users with the necessary resources to quickly engage in problem solving activities, background knowledge in the field is needed as modeling electrical and computer systems is not as intuitive as other modeling activities.

3.2.2 Programming Learning Environments

Educational programming environments are designed to allow novice users to specify algorithms by simplifying general purpose programming languages such as C and Java, which often have steep learning curves for new learners. For example, novice programmers challenged by code organization, the relationship between multiple files within a project, how computers execute code, the use of language-specific keywords that are interpreted differently in natural language as well as the use of Boolean logic expressions. Educational programming environments strive to simplify general purpose programming languages by eliminating unnecessary syntactic details from its users as well as omitting unfamiliar or ambiguous command names. Another common characteristic of these programming environments is the use of graphical objects to represent the components of a program. Syntax of command structures, control structures and variables can be encoded as part of the shapes of graphical objects, limiting connectivity between components when the combination of components do not produce syntactically correct statements [53].

One of the main goals of educational programming environments is to provide students with the computational reasoning and thinking skills necessary to solve problems. Users can develop these solutions (e.g., algorithms) without the roadblocks typically associated with learning new programming languages. How to make programming accessible to precollege learners is a far-reaching area in computer science, a comprehensive review of programming learning environments has been performed by Kelleher et al. [53]. Thus, instead of reiterating the observations and platforms discussed, we present several emerging technologies not previously presented by Kelleher et al. [53].

Scratch [100] is a graphical programming language used by novice programmers to create interactive stories, games, animations, and simulations such as the one provided in Fig. 11a [88]. The *Scratch* language follows three design principles: making it tinkerable, meaningful to its users, and social. Graphic programming blocks are provided such that users snap together blocks to form stacks that represent programs, as illustrated in Fig. 11b. The shape of a block forces a user to connect blocks only in ways that make syntactic sense. Furthermore, users can define several programs, or block stacks, in the Scratch scripting interface. These programs can run individually or as parallel threads by clicking on their graphical representations. Debugging of programs is a straightforward process as users can change blocks as the program is running. To adhere to the meaningful and social design paradigms, users can additionally personalize their projects by uploading pictures and music, recording voices and creating their own graphics. The *Scratch* programming language is also accompanied by a website that supports an online community where users can critique, collaborate and build on other projects. Similar to the *Scratch* users' online community the *ScratchEd* online community is directed toward educators,

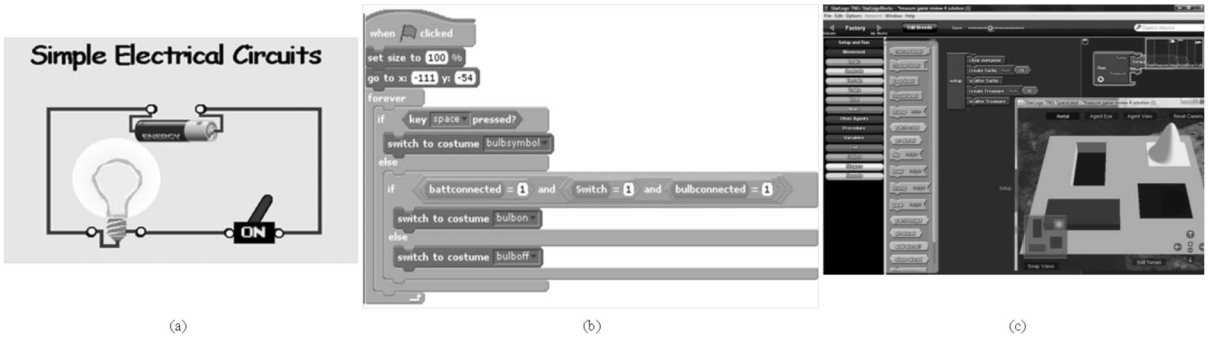

Fig. 11. Programming learning environments. (a) Graphic simulation of an electrical system programmed in *Scratch.* (b) Segment of the *Scratch* code for simulating an electrical system. (c) Programming a 3D world landscape in *StarLogoTNG.*

and provides an online space where educators can share their experiences using Scratch and lesson plans.

Starlogo the Next Generation (TNG) [9] is another example of a visual programming language targeted for middle and high school students (Fig. 10c). *Starlogo TNG's* application domain focuses on the development of 3D computer games. In addition to providing an engaging application scenario to introduce students to programming, the development of 3D computer games offers a rich educational experience requiring users to program the behavior of animated characters, develop storyboards through sequencing and branching similar to binary decisions diagrams, application of art, geometry, and iterative functions. *Starlogo TNG* encompasses two additional modules to aid in the development of games, *Spaceland* and *StalogoBlocks*. *Spaceland* is a 3D OpenGL-based virtual world that can be navigated using a keyboard. Users can view the virtual world as a top down 3D representation or from the viewpoint of one of the game's characters. Users customize the virtual worlds by modifying the existing characters, background, texture, color, and topography of the landscapes. Like *Scratch*, the *StarlogoBlocks* is a graphical programming language composed of building blocks encoded with syntax based on their shapes. Users program three main components to develop their own games: the characters, the scenery, and the behavior. The characters are implemented with *Starlogo* turtles. The types of characters can be specified with the *Starlogo* breeds. The scenery is defined using the *Starlogo-Terrain* editor that supports landscaping, drawing and character placement using onscreen tools to edit the program in real time without typing any code. The behavior of the game and characters within the game is specified using the *StarlogoBlocks* programming interface. Although programs in *Starlogo TNG* are derived from hundreds of different commands and programming lines, the teaching environment presents students with an exciting reason to learn programming while keeping students motivated to overcome any learning difficulty.

3.2.3 Online Resources and Computer Games

Several online resources which provide information regarding engineering, engineering careers, and relevant concepts for students and educators are openly available. The range of learning activities offered by these online resources is typically limited to direct instruction through the presentation of information in different formats such as text, video, and animations, where learners can navigate the site's content as desired. Additional complementary activities include quizzes, puzzles, and games where users can test their knowledge and understanding. While we cannot possible list all these resources, we point out several sites such as *Discovery Engineering* [20], *Engineering your life* [28], *Engineering girl!* [27], and *Packetville* [16].

Moreover the numbers of software technologies that utilize on- and offline computer games to engage students in learning activities are only expected to grow [5]. The development of more educational computer games such the ones discussed by Kelleher et al. [53] is foreseen in engineering education and other related fields.

4 Discussion

A review of hardware- and software-based educational implements have been discussed, offering learning experiences for individuals with various learning preferences and accommodating a diversified number of teaching methods and engineering topics. Some of the tools outlined are well suited to answer one-time impromptu learning demands, while other tools require a considerable investment of time before producing artifacts. As both deductive and inductive methods are essential for precollege engineering education, we argued that some educational technologies are more suitable to teach concepts through deductive methods, like direct instruction, while other technologies facilitate the implementation of the inductive methods of problem/inquiry and project-based lessons. Software-based technologies encompass a rich number of engineering related topics. On the other hand, the most prominent engineering topics covered by hardware-based technologies are related to computer, electrical and mechanical engineering. The development of additional hardware-based tools that encompass a larger number of engineering domains areas are needed to provide student a more diverse learning experience.

While numerous resources are available, challenges remain in the field of engineering education that addressed how to make these learning technologies accessible to a greater number of students. Recognizing the two types of consumers of educational technologies, students and instructors, is of utmost importance. Many of the technologies reviewed in this work have been developed to attract and engage students into engineering related experiences with the instructors' needs playing a secondary role. Developers

of learning technology must remain cognizant that the ultimate facilitators of teaching strategies and tools in the classroom are the elementary, middle, and high school instructors. In addition to the development of learning tools themselves, there is a need to address other factors that might influence precollege engineering instructors to integrate these new technologies into their curricula. Specifically, emphasis should not only be placed on the learning advantages for students and the quality of the artifacts produced with the learning tools, but issues relating to classroom management while introducing and utilizing these technologies. Therefore, empirical evaluations are need to evaluate the effectiveness of learning technologies in precollege engineering education and instructors' perceptions with regard to incorporating these technologies within the classroom. Moreover, the instructional methods to teach precollege engineering must be similarly be considered, mandating the need for technical support and training for instructors, along with predesigned lesson plans that are aligned with the educational standards outlined by district and statewide agencies. We anticipate the increase of initiatives such as *Project Lead the Way* [87] and *Engineering by Design* [26], which strive for the development of precollege engineering curriculum and didactical material. Engineers teaching in K-12 education environments is not common, thus precollege engineering technology should be to empower instructors and students alike in developing sustainable teaching and learning models which cut out the dependencies of expert instructors.

Finally, accessibility to engineering education is a multifaceted problem, with unique restrictions based on individual school and program needs. Access to the human and economical resources needed to offer introductory engineering courses and extracurricular workshops are challenging. Programs such as One-to-one computing that strive to equip each student with one laptop, hold promises for the field [96], but still cannot be considered a nationwide asset in the USA. Whether or not the reviewed technologies in this work are considered as low-cost products within the market, a substantial upfront monetary investment is required to enable students to build interesting and challenging engineering projects, which is out of reach for many institutions. Lower cost educational platforms are needed that can be used for in-depth and elaborated engineering projects independently from the institutions' technological infrastructure, and their human resources technical expertise. Such platforms will have a greater chance to be acquired and prevail in resource-constrained institutions not currently targeted.

REFERENCES

[1] 4M Industrial Development Limited, Green Science Series, http://www.4m-ind.com, 2010.

[2] Aldebaran Robotics, NAO, http://www.aldebaran-robotics.com, 2011.

[3] America COMPETES Act, H.R. 2272, 2007.

[4] American Society for Quality, "Engineering Image Problem Could Fuel Shortage, ASQ Survey: Career Not on Radar for Kids or Parents," Jan. 2009.

[5] F. Arango et al., "A Review of Applications of Computer Games in Education and Training," *Proc. 38th ASEE/IEEE Frontiers in Education Conf.,* 2008.

[6] Arduino, http://www.arduino.cc, 2010.

[7] Autodesk, AutoCAD, http://usa.autodesk.com, 2011.

[8] *How People Learn: Brain, Mind, Experience, and School,* J.D. Bransford, A.L. Brown, and R.R. Cocking, eds., Nat'l Academy, 2000.

[9] A. Begel and E. Kopler, "StarLogo TNG: An Introduction to Game Development," *J. E-Learning,* 2005.

[10] B.S. Bloom, M.D. Englehatt, E.J. Furst, W.H. Hill, and D.R. Kratch, *Taxonomy of Educational Objectives, Handbook 1: Cognitive Domain.* Mckay, 1964.

[11] P.C. Blumenfeld, T. Kempler, and J.S. Krajcik, "Motivation and Cognitive Engagement in Learning Environments," *Cambridge Handbook of the Learning Sciences,* R.K. Sawyer, ed., pp. 475-488, Cambridge Univ., 2006.

[12] Business-Higher Education Forum, "The American Competitiveness Initiative: Addressing the STEM Teacher Shortage and Improving Student Academic Readiness," BHEF 2006 Issue Brief, 2006.

[13] G. Campbell, R. Denes, and C. Morrison, *Access Denied: Race, Ethnicity and the Scientific Enterprise.* Oxford Univ., 2000.

[14] "Cellbots: Using Cellphones as Robotic Control Platforms," http://www.cellbots.com, 2011.

[15] H. Christensen et al., "A Roadmap for US Robotics: From Internet to Robotics," *Computing Community Consortium,* 2009.

[16] Cisco Systems, "Packetville," http://www.cisco.com/web/learning/netacad/packetville, 2011.

[17] Committee on Prospering in the Global Economy of the 21st Century: An Agenda for American Science and Technology, National Academy of Sciences, National Academy of Engineering, Institute of Medicine, *Rising above the Gathering Storm: Energizing and Employing America for a Brighter Economic Future,* http://www.nap.edu/catalog/11463.html, Nat'l Academy, 2007.

[18] R.L. Custer, J.L. Daugherty, and J.P. Meyer, "Formulating a Conceptual Base for Secondary Level Engineering: A Review and Synthesis," *Research in Engineering and Technology Education,* National Center for Engineering and Technology Education, 2009.

[19] Dassault Systems, Solidworks, http://www.solidworks.com, 2011.

[20] Discover Engineering, http://www.discoverengineering.org, 2011.

[21] J. Douglas, E. Iversen, and C. Kalyandurg, "Engineering in the K-12 Classroom. An Analysis of Current Practices and Guidelines for the Future," A Production of the ASEE-Engineering K-12 Center, Nov. 2004.

[22] J. Eccless, "Sex Differences in Achievement Patterns," *Proc. Nebraska Symp. Motivation,* vol. 32, pp 97-132, 1984.

[23] Elenco, Snap Rover, http://www.snapcircuits.net, 2011.

[24] Elenco, Snap-Micro, http://www.snapcircuits.net, 2011.

[25] Elenco, SnapCircuits, http://www.snapcircuits.net, 2011.

[26] Engineering by Design, http://www.iteaconnect.org, 2011.

[27] Engineering Girl!, http://www.engineergirl.org, 2011.

[28] Engineering Your Life, http://www.engineeryourlife.org, 2011.

[29] Engino Play to Invent, Mechanical Science Series, http://www.engino.com, 2011.

[30] European Robotics Technology Platform, Robotics Visions to 2020 and Beyond. The Strategic Research Agenda for Robotics in Europe, 2009.

[31] K. Forbus, K. Carney, B.L. Sherin, and L.C. Ureel II., "VModel: A Visual Qualitative Modeling Environment for Middle-School Students," AI Magazine, vol. 26, no. 3, 2005.

[32] Fischertechnik, "Building Blocks for Life," http://www.fischertechnik.de, 2009.

[33] Gears Educational Systems LLC, "Totally Trebuchet," http://www.gearseds.com, 2011.

[34] J. Glenn et al., "Before It's Too Late," A Report of the Nation from the Nat'l Commission on Math. and Science Teaching for the 21st Century, Dept. of Education, 2000.

[35] I.F. Goodman et al., "Final Report of the Women's Experiences in College Engineering (WECE) Project," Goodman Research Group, Apr. 2002.

[36] Google, "Google Sketchup," http://sketchup.google.com, 2011.

[37] J. Grandy, "Ten Years Trends in SAT Scores and Other Characteristics of High School Seniors Taking the SAT and Planning to Study Mathematics Science and Engineering," Educational Testing Service, Oct. 1987.

[38] S. Greenberg and C. Fitchett, "Phidget: Easy Development of Physical Interfaces through Physical Widgets," *Proc. 14th Ann. ACM Symp. User Interface Software and Technology,* pp. 209-218, 2001.

[39] Handy Board, http://www.handyboard.com/hb, 2009.

[40] E. Machi, "Improving U.S. Competitiveness with K-12 STEM Education and Training," Heritage Foundation, Heritage Special Report, SR-57, June 2009.

[41] Horizon Fuel Cell Technologies, Renewable Energy Kit, http://www.horizonfuelcell.com, 2011.

[42] M. Horn and R. Jacob, "Designing Tangible Programming Language for Classroom Use," *Proc. First Int'l Conf. Tangible and Embedded Interaction,* pp. 159-162, 2007.

[43] Innovation First Int'l, VEX Robotics Design System, http://www.vexrobotics.com, 2009.

[44] H. Ishii and B. Ullmer, "Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms," *Proc. SIGCHI Conf. Human Factors in Computer Systems,* 1997.

[45] I-SOBOT, http://www.isobotrobot.com, 2010.

[46] iRobot, http://store.irobot.com, 2009.

[47] ISEE Systems, STELLA, http://www.iseesystems.com, 2011.

[48] S. Jackson, "The Quiet Crisis: Falling Short in Producing American Science and Technical Talent," Building Eng. and Science Talent (BEST) Report, 2004.

[49] J. Johnson, "Children Robotics and Education," *Artificial Life and Robotics,* vol. 7, pp. 16-21, 2003.

[50] Junun Robotics, Mark III Robot Store, http://www.junun.org, 2009.

[51] K-Team Corporation, Hemisson, http://www.k-team.com, 2006.

[52] *Engineering in K-12 Education: Understanding the Status and Improving the Prospects.* L. Katehi, G. Pearson, and M. Feder, eds., Nat'l Academy of Eng. and Nat'l Research Council, Nat'l Academies Press, 2009.

[53] C. Kelleher and E. Pausch, "Lowering the Barriers to Programming: A Survey of Programming Environments and Language for Novice Programmers," *ACM Computing Surveys,* vol. 37, no. 2, pp. 83-137, 2005.

[54] P. Kirschner, J. Sweller, and R. Clark, "Why Minimal Guidance during Instruction Does Not Work: An Analysis of the Failure of Constructivist, Discovery Problem-Based, Experiential, and Inquiry-Based Teaching," *Educational Psychologist,* vol. 41, no. 2, pp. 75-86, 2006.

[55] D. Klahr, L. Triona, M. Strand-Cary, and S. Siler, "Virtual versus Physical Materials in Early Science Instruction: Transitioning to an Autonomous Tutor for Experimental Design," *Beyond Knowledge: The Legacy of Competence,* Springer, 2008.

[56] I.C. Lee and F.Y. Tsai, "Internet Project Based Learning Environments: The Effects of Thinking Styles on Learning Transfer," *J. Computer Assisted Learning,* vol. 20, pp. 31-39, 2004.

[57] Lego Education, "eLAB Renewable Energy Set," http://www.legoeducation.us, 2011.

[58] Lego Education, "Simple and Motorzied Mechanisms," http://www.legoeducation.us, 2010.

[59] Lego Group, "Mindstorms," http://mindstorms.lego.com, 2009.

[60] M.C. Linn, "Meta-Analysis of Studies of Gender Differences: Implications and Future Directions," *The Psychology of Gender Advances through Meta Analysis,* J.S. Hydes and M.C. Linn, eds., pp. 210-231, Johns Hopkins Univ. Press, 1986.

[61] Logiblocs, "Electronic Discovery System," http://www.logiblocs.com, 2009.

[62] Logiblocs, "LogiRobot Kit," 2011.

[63] Lynxmotion Robotic Kits, http://www.lynxmotion.com, 2009.

[64] R.E. Mayer, "Rote versus Meaningful Learning," *Theory into Practice,* vol. 41, no. 4, pp. 226-232, 2002.

[65] M. McKay and B. McGrath, "Real World Problem Solving Using Real Time Data," *Int'l J. Eng. Education,* vol. 23, no. 1, pp. 36-42, 2007.

[66] Meccano, Super Construction Set, http://www.meccano.com, 2011.

[67] M. Mehalik, Y. Dopplet, and C. Schuun, "Middle School Science through Design Based Learning versus Inquiry: Better Overall Science Concept Learning and Equity Gap Reduction," *J. Eng. Education,* Jan. 2008.

[68] C. Merrill, R. Custer, J. Daugherty, W. Martin, and Z. Young, "Delivering Core Engineering Concepts to Secondary Students," *J. Technology Education,* vol. 20, no. 1, pp. 48-64, 2008.

[69] D. Merrill, J. Kalanithi, and P. Maes, "Siftables: Towards Sensor Networks User Interfaces," *Proc. First Int'l Conf. Tangible Embedded Interaction,* 2007.

[70] K. Michael, "The Effect of a Computer Simulation Activity versus a Hands-On Activity on Product Creativity in Technology Education," *J. Technology Education,* vol. 13, no. 1, pp. 31-43, 2001.

[71] D. Miller, I. Nourbakhsh, and R. Siegwart, "Robots for Education in the Springer Handbook of Robotics," Springer, pp. 1283-1301, 2008.

[72] MIT Media Laboratory, GoGo Board, http://www.gogoboard.org, 2007.

[73] Mobile Robots, Inc., http://www.activrobots.com, 2007.

[74] National Instruments, LabVIEW, http://www.ni.com, 2011.

[75] Nerdkits, LLC, NerdKits, Electronic Education for a Digital Generation, http://www.nerdkits.com/, 2009.

[76] S. Olson and S. Loucks-Horseley, "Committee on the Development of an Addendum to the National Science Education Standards in Scientific Inquiry," Nat'l Research Council, Inquiry and the Nat'l Science Education Standards: A Guide for Teaching and Learning, 2000.

[77] OWI Moon Walker II, http://www.owirobots.com, 2011.

[78] OWI Weasel, http://www.owirobots.com, 2011.

[79] Parallax, Inc., Basic Stamp II, http://www.parallax.com, 2010.

[80] Parallax, Inc., BoeBot, http://www.parallax.com, 2010.

[81] Parallax, Inc., SumoBot, http://www.parallax.com, 2010.

[82] Parallax, Inc., Scribbler, http://www.parallax.com, 2010.

[83] J.C. Perrenet, P.A.J. Bouhuijs, and J.G.M.M. Smits, "The Suitability of Problem Based Learning for Engineering Education: Theory and Practice," *Teaching in Higher Education,* vol. 5, no. 3, pp. 345-358, 2000.

[84] A. Phalke and S. Lysecky, "Adapting the eBlock Platform for Middle School STEM Projects: Initial Platform Usability Testing," *IEEE Trans. Learning Technologies,* vol. 3, no. 2, pp. 132-164, Apr.-June 2010.

[85] Pololu Robotics and Electronics, Pololu 3PI Robot, http://www.pololu.com, 2010.

[86] M. Prince and R. Felder, "Inductive Teaching and Learning Methods: Definitions, Comparisons and Research Bases," *J. Eng. Education,* vol. 95, pp. 123-138, Apr. 2006.

[87] Project Lead the Way, http://www.pltw.org, 2011.

[88] M. Resnick, "Behavior Construction Kits," *Comm. ACM,* vol. 36, no. 7, pp. 64-71, 1993.

[89] Revolution Education Ltd. Chip Factory, http://www.rev-ed.co.uk, 2010.

[90] Revolution Education Ltd. PICAXE, http://www.rev-ed.co.uk, 2009.

[91] Robobuilder, Co., Ltd. ROBOBuilder, http://www.robobuilder.net/, 2009.

[92] Robophilo, http://www.robophilo.com, 2010.

[93] Roboni-i, Action Games, http://www.roboni-i.com, 2010.

[94] Robot Gray Matter, LLC, BRAIN, Your Robot's Intelligence, http://www.robotgraymatter.com, 2009.

[95] Robotech, Edutainment Robots and More... Robo Designer, http://www.robotechsrl.com, 2011.

[96] M. Russell, D. Bebell, and J. Higgins, "Laptop Learning: A Comparison of Teaching and Learning in Upper Elementary Classrooms Equipped with Shared Carts of Laptops and Permanent 1:1 Laptop," *J. Educational Computing Research,* vol. 30, no. 4, pp. 313-330, 2004.

[97] P. Sadler, H. Coyle, and M. Schwartz, "Engineering Competitions in Middle School Classroom," *J. Learning Sciences,* vol. 9, no. 3, pp. 299-327, 2000.

[98] R.J. Savery and T.M. Duffy, "Problem Based Learning: An Instructional Model and Its Constructivist Framework," *Educational Technology,* vol. 35, pp. 31-38, 1995.

[99] D. Schaffer and P. Gee, "Before Every Child is Left Behind: How Epistemic Games Can Solve the Coming Crisis in Education," *Wisconsin Center for Education Research (WCER),* Univ. of Wisconsin-Madison and Academic Advanced Distributed Learning Co-Laboratory, 2005.

[100] Scratch, Imaging, Program Share, http://scratch.mit.edu, 2010.

[101] Solarbotics Ltd, http://www.solarbotics.com/, 2009.

[102] R.J. Spiro and M. DeSchryver, "Constructivism: When It's the Wrong Idea and When It's the Only Idea," *Constuctivist Instruction,* S. Tobias and T.M. Duffy, eds., pp. 106-123, Taylor and Francis Group, 2009.

[103] STEM Education Coordination Act of 2009, H.R.1709, 2009.
[104] J. Sullivan, "The Bridge: A Call for K-16 Engineering Education," *Nat'l Academy of Eng. Nat'l Academies,* vol. 36, no. 2, pp. 17-24, 2006.
[105] Survey Corporation, "Surveyor SRV-1 Open Source Mobile Robot," http://www.surveyor.com/SRV_info.html, 2009.
[106] Business Roundtable, "Tapping America's Potential: The Education for Innovation Initiative,"report, July 2005.
[107] M. Teitelbaum, "The US Science and Engineering Workforce: An Unconventional Portrait," Nat'l Academies' Govt. Univ. Industry Research Roundtable (GUIRR) Summit, 2002.
[108] Thames and Kosmos Science Kits, http://www.thamesandkosmos.com, 2009.
[109] "PicoBoard, Connect Real-World Sensors to Your Scratch Project," http://www.picocricket.com/picoboard.html, 2010.
[110] W.J. Thomas, "A Review of Research on Project-Based Learning," *The Autodesk Foundation,* Mar. 2000.
[111] Tribotix, http://www.tribotix.com, 2009.
[112] C.R. Warren, "An Exploration of Factors Influencing the Career Preferences of Junior High Students," Ann. Meeting of the Nat'l Science Teachers Assoc., 1990.
[113] K. Welde, S. Laursen, and H. Thiry, "Women in Science, Technology, Engineering and Math (STEM)," *Sociologist for Women in Soc.,* fact sheet, http://www.socwomen.org/socactivism/stem_fact_sheet.pdf, 2007.
[114] White Box Robotics, http://www.whiteboxrobotics.com, 2009.
[115] G. Wimberly and R. Noeth, "College Readiness Begins in Middle School," ACT Policy Report, Act, Inc., 2005.
[116] WowWee Astonishing Imagination, "Rovio," http://www.wowwee.com, 2010.
[117] WowWee Astonishing Imagination, "RS-Media," http://www.wowwee.com, 2010.
[118] P. Wyeth and G. Wyeth, "Electronic Blocks: Tangible Programming Elements for Preschoolers," *Proc. Eighth IFIP TC13 Conf. Human-Computer Interaction,* 2001.
[119] Yost Engineering, "BugBrain," http://tech.yostengineering.com, 2011.
[120] J.G. Zubia, "Educational Software for Digital Electronics; BOOLE DEUSTO," *Proc. Microelectronic Systems Education,* pp. 20-22, 2003.

Mario Riojas received the BS degree in computer engineering from CETYS University, Mexico, and the MS degree in electrical and computer engineering from The University of Arizona. Currently, he is working toward the doctoral degree in electrical and computer engineering at The University of Arizona. His active research is on precollege engineering education, ubiquitous computing, human-computer interaction, and technological literacy. He is a member of the IEEE.

Susan Lysecky received both the MS and PhD degrees in computer science from the University of California, Riverside, in 2003 and 2006, respectively. Currently, she is working as an assistant professor in the Department of Electrical and Computer Engineering at the University of Arizona. She coordinates research efforts for the Ubiquitous and Embedded Computing lab, and her current research interests include embedded system design with emphasis on self-configuring architectures, human-computer interaction, and facilitating the design and use of complex sensor-based system by nonengineers. She is a member of the IEEE and ACM.

Jerzy Rozenblit received the PhD degree from Wayne State University, Michigan, and the MS degree in computer engineering from The Technical University of Wroclaw, Poland. Currently, he is working as a professor of electrical and computer engineering at The University of Arizona. He serves as an associate editor of the *ACM Transactions on Modeling and Computer Simulation*, associate editor of the *IEEE Transactions on Systems, Man and Cybernetics*, and a reviewer for a number of national and international funding agencies. In 1994/1995, he was Fulbright senior scholar and visiting professor at the Institute of Systems Science, Johannes Kepler University, Austria. He has also held visiting scientist appointments at the Central Research Laboratories of Siemens AG, Munich. His research and teaching include complex systems design and simulation modeling. His research in design has been supported by the US National Science Foundation, Siemens AG, Semiconductor Research Corporation, McDonnell Douglas, and the US Army Research Laboratories, where he was a research fellow. He is a senior member of the IEEE.