A Multi-Pronged Approach to Bringing Embedded Systems into Undergraduate Education[1]

Betty H. C. Cheng, Diane T. Rover, and Matt W. Mutka

Department of Computer Science and Department of Electrical Engineering
Michigan State University
East Lansing, MI 48824

e-mail: {chengb,mutka}@cps.msu.edu, rover@egr.msu.edu

Abstract

Embedded computer systems play an increasingly important role in today's society. Such diverse technologies as avionics, automobile drive trains, communication systems, and medical equipment are relying on computers to control system parameters. Although embedded computers are powerful and flexible tools for industry, these very advantages have contributed to a corresponding increase in system complexity. In order to adequately prepare today's computer science, computer engineering, and electrical engineering students for their future careers, the special problems with embedded systems development must be adequately addressed in their education. In this paper, we report on our multi-pronged approach to curriculum development that specifically incorporates embedded systems into a suite of relevant courses: software engineering, operating systems, digital system design, and computer system design (capstone course in computer engineering). The approach comprises modular course pack development, suitable for alternative teaching models, such as team teaching and development of multidisciplinary courses; team projects to give students hands-on experience with embedded systems; and incorporation of innovative teaching techniques designed to facilitate and enhance the student's learning experience.

Curriculum developments focus on embedded systems and our courses. In the software engineering course, students are exploring how object-oriented development techniques can be applied to industry-oriented embedded system projects, such as process controllers for numerous appliances (e.g., washing machines and dishwashers) and automotive-related systems (e.g., climate control and door controls). The operating system course includes a new module in real-time scheduling, with laboratory projects planned for task allocation in real-time distributed computing systems. Our digital system design course sequence includes the design of application-specific integrated

'This work is sponsored in part by NSF grants CDA-9700732, CDA-9617310, CCR-9633391, CCR-947318, CDA-9529488,ASC-9624149, MIP-9321255.

circuits using VLSI and programmable logic, with plans to emphasize their role in embedded systems. The computer engineering capstone course provides a cooperative learning, cross-functional teaming approach that studies the tradeoffs of hardware-software co-design in embedded control applications; for example, a fluid flow problem is formulated via controlling the movement of a floating ping-pong ball in a tube by adjusting air flow. The learning model includes formal skill teams that share specific skills and separate design teams to attack specific design problems. We believe the curriculum developments improve both the instruction and practice of computer science and engineering, to provide students with greater insight about the functionality, reliability, and other properties of an embedded system.

1 Introduction

This paper presents a multi-pronged, multidisciplinary approach to transfer research results and experience into the undergraduate curricula in the Departments of Computer Science and Electrical Engineering at Michigan State University. The objective of the project is to explore the impact of embedded systems in a number of core courses in the undergraduate curricula. First, existing courses have been modified to explicitly present the respective course content in the context of embedded systems. Specifically, four courses are targeted in the VESL Visions for Embedded Systems Laboratories) project: software engineering, operating systems, computer system design, and digital electronics. The courses were selected based on a multidisciplinary approach to exploring how different facets of an embedded system are developed. Second, course modules and laboratory exercises were developed to give students hands-on experience with embedded systems for a given topic of study, for example, operating systems and digital logic. Third, several innovative teaching techniques were developed and applied to facilitate the understanding of the course materials in the context of embedded systems. Finally, two of the courses involved group projects in the application of new research techniques as applied to embedded systems.

The remainder of this paper is organized as follows. Section 2 describes the VESL project with respect to its short term and long term objectives. Section 3 describes how specific existing courses were modified in order to address VESL project objectives. Section 4 describes the group projects that students developed in two separate courses. A summary of the complementary innovative techniques that were explored by the instructors to enhance the learning experience of the students is given in Section 5. Finally, conclusions and future investigations are discussed in Section 6. Also included in this last section is a summary of the tangible products being developed for the VESL project for the related courses that may be used by other institutions interested in exploring similar technology transfer opportunities.

2 Overview of VESL Project

VESL is a project that is designed to address three needs in undergraduate computer science and engineering education: (1) providing students with hands-on, real-world experiences in embedded computer system development in an active, collaborative learning environment; (2) increasing accessibility of specialized laboratories to students to allow different learning paces and approaches and to support different contexts and types of interaction; and (3) sharing costly laboratory resources with other universities [l]. In order to address these needs, we have initiated a three-pronged strategy: (1) instructional module development, (2) laboratory development, and (3) design project development.

The VESL project is being performed at Michigan State University in the context of a newly revised undergraduate Computer Engineering Program within the Departments of Electrical Engineering and Computer Science and in collaboration with three smaller universities in the State of Michigan: Lake Superior State University, Grand Valley State University, and Saginaw Valley State University. These universities represent several types of engineering programs (from an electrical/computer engineering technology program to an electrical engineering program) which include an embedded system design component. The VESL name highlights an important aspect of the curriculum development: use of laboratories to support integration of material throughout a student's program. Moreover, the name is symbolic of the connective and open features of the proposed laboratory. It is intended to support connective learning, facilitating closer interaction among students and instructor. It is designed to connect students with real systems and real problems. The course materials resulting from this project will be accessible to students on and off campus via the Internet, so that "open" takes on an extended meaning encompassing accessibility, networked/interoperable systems, distance education, and resource sharing.

The focus for the curriculum development is embedded computing systems. Research results in real-time computing, software engineering, and wireless systems are being transferred to the instructional domain. We are applying a concept-driven, modular development approach that is called *"Open Your I's"* (to active learning). The premise of the approach is that concepts are learned by *a* process of *Introduction, Instruction, Illustration, Investigation,* and *Implementation.* For example, concept illustration refers to live or multimedia demonstrations in the classroom; concept investigation refers to hands-on or multimedia interactive exercises; and concept implementation refers to hands-on laboratory projects. The development of "I" modules will facilitate integration across the curriculum, including cross-pollination of subject matter, as well as dissemination to other universities. Additionally, it provides a structured means to transfer techniques and tools from research into instruction.

The curriculum development activities of the VESL project are intended to improve both the instruction and practice of computer science and engineering. Embedded systems are inherently multidisciplinary, meeting the computing needs of scientists and engineers in many domains. Realistic design experiences provide the student with greater insight about

the functionality, reliability, and other properties of a system, allowing the system to better serve its domain-specific requirements.

2.1 Goals and Objectives

The following is a list of objectives of the embedded systems curriculum development activities highlighted in this paper:

Address engineering education needs: The VESL project addresses needs identified in recent discussions on engineering education [2, 3, 4]. The objectives that follow reflect particular needs that will be impacted by the project and their significance. Several innovations of the VESL project enable it to have local as well as national impact: the Open Your I's paradigm and modules; the open laboratory concept and multi-institution collaboration; and the backdrop of a restructured Computer Engineering Program that is committed to providing a rich learning environment, a focus on embedded hardware/software systems development, and integrated, realistic design experiences with industry involvement.

Address industry needs: The exercises and projects (1) provide students with fundamental knowledge in computer hardware and software co-design so that they can adapt to rapidly changing computer technology, and (2) familiarize students with state-of-the-art technologies so that they possess necessary skills to contribute to the computer industry [5]. Additionally, the embedded systems focus provides students with the technical depth that industry seeks [6]. However, we are careful not to abandon the fundamentals of our field to make room for the ever-changing needs of industry [7].

Provide hands on experience under realistic conditions: We are using laboratory exercises and projects that integrate all aspects of system development. The intent is to make our students aware that engineering does not happen in a vacuum and that, in order to develop a successful product, the engineer must interact with many non-engineers. For example, we plan to use case histories and case studies developed in cooperation with industrial partners [6].

3 Renovations of Existing Courses

This section overviews three courses that have been reviewed and revised to be taught in the context of embedded systems. Special emphasis is placed on incorporating recent results from embedded systems research. The courses are described in the order that corresponds to

a top to bottom view of the development of embedded systems. That is, we start at a high-level by describing the software engineering course that addresses how embedded systems are defined and requirements are analyzed. Next, we describe how the operating systems course presents system software and services relevant to embedded systems, particularly real-time operating system (RTOS) concepts. Finally, we describe the digital system design courses that investigate how components of embedded systems are designed, implemented, integrated, and tested, ranging from microprocessors, to integrated circuits, to interfaces.

3.1 Application of Software Engineering to Embedded Systems

The software engineering (SE) course is designed to teach students the fundamentals of software development, beginning with problem identification and requirements analysis through design and testing. Significant emphasis is placed on object-oriented analysis and design techniques.

In general, a given system should be developed by undergoing a requirements analysis, high-level design, detailed design, coding, testing, and development of a user manual. Initially, we had only one software engineering course at Michigan State University. Students learned and applied the fundamentals of software engineering to group projects. Clearly, this was a challenging task. Based on student feedback, it was clear that the first two stages of development were the most challenging for the students. Students frequently remarked that they wish that they had an opportunity to rework those particular documents. Recently, the single SE course has been divided into two courses. The first course introduces students to the fundamentals of SE, and the second course is a senior capstone course, where students apply SE on a group project sponsored by industry. While the students do not implement a project in the first SE course, the students do work in teams on three deliverables for a real project, namely, the requirements analysis document, including a prototype, the high-level design document, and a user manual. Therefore, the students have at least two opportunities to work on the two critical stages of software development, namely requirements analysis and design.

Previously, software engineering education has been largely directed towards mainstream software-based applications, such as the development of client-server information systems, network management systems, and CASE (Computer-Aided Software Engineering) tools. Most of the techniques taught in the software engineering courses are generically applicable to software-based systems, with little emphasis placed on hardware environments. In order to complement the current material presented in software engineering courses and to provide training in an area that is gaining increasing attention, we extended the SE course to cover software development techniques specific to embedded systems in addition to those that are generally applicable to software systems.

The course has had a focus on object-oriented development techniques. But we have found

that, traditionally, embedded systems have not been developed within the object-oriented paradigm. And even within the structured analysis and design paradigm, there seemed to be a lack of emphasis on a stepwise refinement process. One of our current research projects is to explore how a commonly used object-oriented modeling technique (OMT [8]) can be used to model embedded systems [9] and how the models can be used as the basis of formal specifications [l0, 11, 12]. The formal specifications enable developers to perform numerous analysis tasks, such as checks for completeness and consistency of the diagrams, which would otherwise only be analyzable by visual inspection. Therefore, the formalization of OMT bridges the gap between the graphical, easy to use notation that might be ambiguous and prone to errors with formal specifications that enable rigorous analysis and verification activities.

In order to use OMT to model embedded systems, we had to introduce new guidelines for model development. OMT comprises three complementary modeling techniques, constructed in the following order. First, the object model describes the static, structural aspects of the system, that is, the objects and the relationships between the objects; this model is similar, in form, to the entity-relation diagrams traditionally used to model database systems. Second, the dynamic model is used to describe the states of the system and the events that cause the system to transition between the states; the state diagram is used for the dynamic model. Finally, the functional model is used to describe the services and the data flow of the system; the data flow diagram (DFD) is used to describe this model. We found that for all embedded systems, the object model could be defined uniformly to comprise a controller, sensors, and actuators. The relationships between these three main types of objects are typically straightforward to define. In contrast, the state diagram (dynamic model) for an embedded system can become quite complex given the potentially large number of combinations of inputs from sensors and the number of possible responses to be effected by the actuators. The functional model of the system may or may not be complex, depending on the specific embedded system and the number of services that it provides.

In addition, students were taught how to identify safety-critical aspects of the system and how to use logic to formally specify these critical conditions. We define the term ‘safety-critical’ conditions to refer to those constraints that if violated may cause something ‘bad’ to happen, that is, loss of life, property, or money. This exercise greatly helped to demonstrate to the students the value of the process of formally specifying critical conditions. Students found that English descriptions could be ambiguous and incomplete, and sometimes even contradictory. By going through the process of representing the conditions in logic, students detected these inconsistencies and ambiguities, even without tool support. These findings are in line with what our experience has been with industrial collaborators.

3.2 Application of Operating Systems to Embedded Systems

Principles of operating systems (OS) have a long tradition of being required instruction for computer science and engineering majors at many colleges and universities, including Michigan State University. Since many of the new engineering problems to be solved in industry are problems of resource management and allocation, timing constraints, and limitations in memory and power for embedded systems, we want to bring new modules to our operating systems course to reflect these situations.

We are beginning to integrate issues in embedded computing systems into the operating systems course by focusing on real-time systems. Since scheduling is the key for the design of real-time embedded systems, our undergraduate computer science and engineering students are being instructed in the mature results from research of real-time scheduling, including algorithms such as rate monotonic, earliest deadline first, and least-laxity first. In particular, they are being taught the concepts of feasible scheduling of hard real-time systems, and the difference between hard and soft real-time systems. The primary difference between these systems is whether a missed deadline of a task in a system means a failure of a system (hard real-time), or whether an occasional miss of a deadline is acceptable as long as the frequency that the system misses the deadline is low (soft real-time). For example, the sensor control loop of a robot positioning task might need to operating at 1 KHz. If the task ever misses its deadline, then the robot arm will fail, meaning the sensor control task has a hard deadline. In contrast, the playback of a video may be expected to occur at 30 frames a second. If the task that decodes a frame and puts the frame in a frame buffer is occasionally late, then the system status remains acceptable. However, if the task is late repeatedly, the system status is failure.

In order to reinforce concepts of real-time scheduling, we are introducing students to tools in the set of laboratory assignments. The tools were built as products of research of real-time systems. These tools include a task allocation system, called DRMS [13], a complex system simulator, called CSS, and a performance visualization system, called VIE (Visualization and Interaction Environment). Students will use these tools in a laboratory experiment to be able to answer interesting questions of resource allocation of a real-time system, similarly as it is done in industry. For example, given a set of critical tasks that must meet their deadlines and a set of tasks that are less critical, but have timing constraints, we want students to deal with questions such as:

- What happens to a feasible set of tasks if the deadline of one task is shortened in the task set?
- What happens if one task is moved to a different processor? Will the task set remain feasible?
- Will the task set remain feasible if the amount of memory allocated to the processor in the system is changed?

The tools introduced in the laboratory enable students to find answers to such questions. In the laboratory, students may specify a complex system, including processors with specific memory sizes, relative processing rates, network interfaces to other processor systems, and message transfer rates for the network interface. Likewise, students specify the frequency that tasks are to be executed, including the memory/processing requirements of the tasks.

The DRMS system takes input of tasks and processor specifications, and it performs a computation based on a real-time scheduling test, known as rate monotonic scheduling, to specify on which processors tasks should be placed so that all tasks may be feasibly scheduled. Then, a priority is assigned to the tasks allocated on the processors. DRMS can perform three rate-monotonic schedulability tests to determine if a task allocation can meet its deadlines, ranging from the most pessimistic, single inequality test, to multiple inequality tests, to the most optimistic branch-and-bound algorithm for task allocation.

Similarly, the Complex Systems Simulator (CSS) takes the specifications and generates simulation modules. The CSS is based on C++SIM, an extensible and portable discrete event simulation library. Simulation entities are objects derived from one of several available thread types, introduced as C++ classes. A direct approach to simulation of computer systems has been taken by letting the simulation entities represent models of real-world hardware and software entities. For example, a processor entity simulates computer hardware running modeled OS services, a router entity simulates communication hardware and software, while a task entity simulates a software component of a modeled parallel or distributed application.

The simulation system may be monitored with instrumentation procedure calls, which generate performance trace records for processing by an instrumentation system manager. The manager then forwards the data to tools for displaying views that depict the performance and execution of the simulator. The students can examine visually the execution of the simulation system and note where real-time constraints are met and missed.

3.3 Application of Digital System Design to Embedded Systems

Three courses represent the extent to which the embedded system theme supports the curriculum in digital system design. In the introductory digital logic fundamentals course, students are motivated to appreciate the role of digital logic in their daily life through a group assignment to identify a product or process driven by digital logic (e.g., traffic lights, digital camera, calculator, microwave oven, etc.). They select, investigate, and discuss a product and write a web-based, electronic essay. Thus, while students are learning the basics, they begin to see the relevance in something tangible. It raises their awareness about computer-based systems as well as raises their curiosity. Heightened student interest improves learning of the fundamentals and provides a basis for subsequent courses in digital system design.

In the senior-level digital electronics course, students learn in-depth about digital circuits,

VLSI design, and application-specific integrated circuits (ASICs). Students are instructed that ASICs are an important hardware component of embedded systems [14]. Students are informed of related topics such as hardware-software co-design, systems-on-a-chip, and IP (intellectual property) cores. Student design groups must address these topics in their projects.

The capstone course on computer system design is the major engineering design experience for seniors in computer engineering [15]. This course has been recently revised, with significant innovations in its learning model [16]. This same model has also been applied in a related capstone course in computer science. The two courses involved, respectively, the design of an embedded system, and the development of web-based interfaces to embedded systems and distributed embedded systems.

In the revised computer system design capstone course, students learned about embedded systems, i.e., electrical systems that contain embedded computers to control processes. Each student actively participated as a member of an engineering design team and made significant contributions to achieving the team's goals. One of the first team projects in the course has involved a ping-pong ball system with sensors and actuators to emulate a fluid-flow control system-a ping-pong ball in a tube is controlled by airflow through a fan and its position is sensed. Projects have centered on use of tools and technologies such as LabVIEW, in-system programmable logic, 68HCll microcontrollers, StateCharts, etc. In addition, projects have involved the use of the web for remote monitoring and control of the embedded system, thus broadening the scope to distributed systems and requiring more advanced development strategies such as hybrid prototyping.

Previously, the course had been taught in a lecture-laboratory format with three 50-minute lectures and one 3-hour laboratory per week. Students were paired in the laboratory and completed a series of pre-defined projects. The content was typical of traditional computer interfacing courses. However, in light of ABET Engineering Criteria 2000, new course learning objectives were developed to be compatible with the computer engineering program's educational objectives, with things learned through a benchmarking process of major engineering design courses, and with input received by the Employer Stakeholder Focus Group-a set of key employers of our graduates enlisted to advise the faculty as program educational objectives and assessment strategies were developed.

The course learning objectives include both team and individual objectives. Teams are required to: propose an engineering design project that has clearly stated design criteria, including realistic constraints; share in the day-to-day design activities and management of the project; share in the presentation of oral and written progress reports; share in the demonstration of results at key milestones during the life of the project; and evaluate the project's progress and outcomes against a clearly articulated set of criteria.

Moreover, individual students build a number of technical, professional, and communication skills, summarized from the complete list of learning objectives [15]: describe a generic em-

bedded system; understand the need for hardware and software standards; delineate design criteria and constraints; describe the overall engineering design process and apply key tools; describe contemporary industry practices; and acquire information from technical literature and the Web.

To realize these objectives, student-learning and course-management concepts were applied [17, 18, 19, 20, 21, 22, 23], resulting in an approach referred to as "cross-functional teaming." In this approach, students are grouped into two sets of interdependent teams, "design teams" and "skill teams." Design teams are formed for the entire semester. Each of these teams works on a specific engineering design project that involves the collaborative development and evaluation of a "product" that contains an embedded computer. Skill teams are formed from representatives of each design team. As the name implies, these teams learn specific skills needed to ensure success within the individual design projects. Skill teams are highly focused, and the intent is to foster self-directed learning in the interests of lifelong-learning as well as learning by teaching others (since skills brought back to design teams must be shared with other members).

In the revised course, learning objectives are achieved through a variety of means in addition to the team design projects. Course modules developed by the instructors provided technical depth where needed, for example, about programmable logic, computer interfacing, microcontroller development tools, etc. The most successful lessons involved cooperative learning and guest speakers. The students read about embedded systems from contemporary technical literature and discussed these in groups in class. One article that has been cited by students as notable focuses on the creativity required in engineering of embedded systems and the hardware-software design decisions bounded by constraints [24]. Students also read about engineering ethics, including ethical considerations in hardware and software quality. Class discussion of ethics case studies involving embedded systems provided students with realistic scenarios and different perspectives. Computer and communication standards and their importance is another discussion topic, highlighting the many standards that students encountered in their embedded system design projects. This topic is reinforced with a guest speaker, who has worked with the IEEE standards development process. This speaker and several others who shared their expertise with the students made a significant impression on students: a specialist (who is blind) from MSU's Office of Programs for Handicapper Students talked about accessibility and designing products to be handicapper friendly; and engineers from industry presented topics, such as requirements-driven design and intellectual property, from their own first-hand experiences. Embedded systems have provided a natural context to present many contemporary and professional issues.

4 Team Projects

This section describes team projects developed by two courses: the software engineering course and the computer system design course. The first course focused on requirements analysis and design, as well as building prototypes to facilitate the requirements analysis stage. Each team worked independently and had a specific customer. The second course is a senior capstone course that developed multidisciplinary projects with an emphasis on team management and cooperative learning between teams. The customers were from other disciplines, thus providing an additional dimension to the students' learning experience.

Software Engineering Course. In order to reinforce the software engineering and object-oriented modeling concepts, we had students work in teams of four working on real-life embedded systems with an emphasis placed on project documentation. These embedded systems were largely obtained from industrial contacts in two main domains: appliances and automotive systems. With these realistic projects, students seemed to be able to better understand the motivation for embedded systems and their differences from more software-based systems. Students also gained a much better appreciation of the complexities of every-day appliances and commonly used automotive features. Projects [25] included controllers for washing machines, dishwashers, ovens, ABS brakes, climate control system, driver notification system, cruise control, and a monorail system. For each project, students were given brief (2-3 page) project descriptions. Based on that information, students were expected to perform domain research to better understand the scope of their respective projects. For each project, there was an identified customer. The students needed to gain approval of the requirements from the customer before proceeding to the design stage. Prototypes were built (using special software available on the SGIs in the Cooperative Multimedia Laboratory previously sponsored by an NSF/ILI grant) in order to facilitate the requirements analysis stage. Students were given six weeks to complete the requirements analysis document. The extended time was allotted in order for the students to fully comprehend, as a team, the intricacies of solicitation, capture, and analysis of requirements. Based on feedback from the requirements analysis document, the students proceeded to produce a high-level design of their systems, which involved the creation of an architectural framework for the system, explicit allocation of hardware and software components, refinement of system components, detailed descriptions of behavior, and communication mechanisms between different parts of the system.

The course concluded by having students give oral presentations and live demonstrations of their respective projects. The students found it quite useful to learn how to give concise and team-choreographed briefings of their projects to their peers. (Most students had not previously given oral presentations.) As an additional exposure to industry practices, students were taught how to use commonly used presentation software (Microsoft's Powerpoint) with an LCD projector, and an SGI was connected to the Internet and was available for the

students to give live demonstrations following their oral presentations.

Therefore, the projects served as an extremely useful vehicle for teaching students how to apply leading-edge software development techniques to real-life applications. It is commonly recognized that a student's communication skills, as observed through written documentation and oral communication, are critical in industry.

Course Evaluation. The project and student evaluation comprised several components. The students turned in weekly status reports that described the accomplishments for the current week, plans for the following week, and any issues to be considered by the instructor. The students were also required to meet on a weekly basis, where the agenda had to be mailed out 24 hours in advance. The agenda and meeting summaries were included in the status reports. After each deliverable was submitted, each student filled out a peer-review form that described each student's perception of the effort expended by each of the team members, including the student. Finally, each deliverable was evaluated by the customer and the instructor. The students played specific roles on the project: project manager (instructor selected), project facilitator, documentation manager, and research coordinator.

Computer System Design Course. Teams were put into a context of a single company's engineering staff meeting a customer's needs. The company, Spartan Embedded Technologies, issued a request-for-proposals (RFP) and the design teams submitted written proposals for design projects in response to the RFP and gave presentations. During the term, teams spent considerable time on written and oral communication, including progress reports, technical reports, final reports and demonstrations, and web sites. Students gave all presentations using Powerpoint, a projection system, and a notebook computer in the classroom. Design projects were taken from requirements to implementation, either by designing a new system or by re-engineering an existing one. In some cases, students from previous terms acted as consultants on existing products.

Design projects contained a number of common threads that facilitated project success through the use of a course learning model emphasizing cooperative learning and cross-functional teaming. K. A. Smith and A. A. Waller have described the principle of "cooperative learning," which focuses on the use of small instructional groups that allow students to work together to maximize their own and each others' learning [18]. In addition, E. Aronson, et al., have described a "cooperative jigsaw strategy" [19], an advanced application of cooperative learning. Merging these two concepts, we have developed an effective organization for cross-functional teaming. We formed base groups, or teams, for the semester that reflected the overall goals of the course- i.e., the course learning objectives. These were the design teams of Spartan Embedded Technologies. However, these teams were not isolated from each other because specialized teams were formed from representatives of each base team. Members learned specialized skills or gained knowledge needed by their base teams,

and shared these with their base teams in support of base team's goals. These specialized teams were the skill teams.

Furthermore, E. Nuhfer [20] has discussed the importance of "student management teams," which can be viewed as a specialized team in the cooperative jigsaw strategy. This team, assigned project management responsibilities, embodied the idea that students themselves should make informed decisions about team activities and assess team performance. With this learning model in the computer system design course, the teacher's role as a stand-up lecturer was diminished, and instead was, as suggested by E. Mazur, one of listening and questioning [21]. By asking design teams and skill teams the right questions at the right time, the instructor engaged students in the learning process toward meeting course objectives.

More specifically, in our implementation of these concepts, students were grouped into design teams and skill teams. Several skill teams were formed regardless of project specifics: course and project management; web site development; and documentation and presentation. A student management team supported overall course coordination, project planning using Microsoft's Project software, and design-team and skill-team assessment; a web site development team built and maintained a web site for each design team; and a presentation and documentation team mastered Microsoft's Powerpoint and Word software and developed standard formats for use by all teams in the company. Each design team had one member on the management, web, and documentation skill teams. These skill teams lasted for the duration of the semester.

Other skill teams have varied from term to term, depending on the nature of the design projects. For example, skill teams focused on the following topics, among others: C and assembly language programming; LabVIEW programming; web programming; software engineering; programmable logic; and sensors and actuators. Representation on these skill teams was determined according to design-team needs. These teams existed only as long as needed to accomplish a designated purpose. Students also proposed and formed new skill teams as the need arose, i.e., students showed responsibility for creating and managing their own learning. The interaction facilitated by skill teams that crossed design-team boundaries was a key aspect of the teaming experiences gained by the students.

The cross-functional teaming model is intended to support multidisciplinary teams. We have thus far involved only faculty and student facilitators from mechanical engineering in the design projects, as logistics are not yet in place to build teams including students from other disciplines. Nonetheless, embedded systems have provided a natural context for evolving multidisciplinary projects.

Course Evaluation. Student performance in the course was evaluated based on meeting team and individual learning objectives. Both team and individual deliverables were submitted regularly. Individual students maintained a journal, in which they entered impressions of the course and identified how they were fulfilling the learning objectives. At the end of

the term, each student wrote a summary "professional self-assessment report." In this report, they assessed their learning in the course, the impact of this course, and their career plans. Individual accountability was reinforced through demonstrations, presentations, in-class discussion, and individual technical assignments (e.g., an application note). The Web was used to help track student and team progress. Students were introduced to strategies for effective teaming, group processing, and self-assessment, including the periodic use of evaluation forms for the team leader, each member, and the team as a whole [23].

5 Technology Interchange and Transfer

Software Engineering Course. Several activities were performed to explicitly address technology transfer and interchange. First, the web was used extensively to foster communication between each group and the instructor, the teaching assistants, and the customer. The software engineering course had a web page that provided access to all course materials, including syllabus, weekly schedule of reading, homework, and lab assignments, as well as all deliverable templates. In addition, all homework and laboratory assignments were available through the course web page, as well as any handouts relevant to the laboratory assignments. (All course lecture materials were made available to the students prior to the start of the semester.) The extensive use of the web decreased the amount of paper flow for the students. Also, the students found it useful to have access to the homework and lab assignments at any time, rather than managing a large collection of papers. Also, if the students missed a class or a lab, it was easy for them to make up their work.

The students also created web pages for each project. The project web pages were required to contain links to each of the deliverables. Some students also created links to the meeting agendas and summaries. Many of the students were quite creative in the design of their homepages. The course web pages enabled convenient access to the project components by all team members, instructor, and the customers. In addition, the project web pages enabled the students to showcase their projects to prospective employers. As an extra option, most of the students opted to create a Java version of their prototypes so that their prototypes would also be available via the web (the SGI prototypes could only be executed on SGIs).

Second, course modules have been developed that can be used by other instructors to teach software engineering and object-oriented development of embedded systems to other classes. The motivation for developing the course modules was twofold. First, the VESL group is investigating the development of a new embedded systems course that would draw from expertise in computer science, electrical engineering, and computer engineering. Topics that will be included in this course are software engineering of embedded systems, hardware-software co-design, real-time operating systems, and signal processing. Since the course provides a broad spectrum of topics, the self-contained course modules enable instructors who might be more familiar with other topics to also present the materials in the modules.

The modules contain complete lecture materials, as well as homework and lab assignments. In addition, the course modules could be presented in the context of a short course for industrial organizations. Or the course modules may be used by other institutions who are also exploring the incorporation of embedded systems into their curricula.

Computer System Design Course. As in the software engineering course, the Web played a key role in sharing and transferring course "intellectual property" among students, instructors, course consultants, project "customers," and employer groups. It is critical for students to develop useful results in an appropriate form that can be effectively conveyed and transferred to different audiences. Throughout the course, the learning model supported this development and dissemination of student work and course/project information.

Students have enthusiastically embraced the course, citing both the teaming and design projects as being representative of real-world industry experiences. Feedback has been obtained through course evaluations, journals, and assessment reports. Student feedback has emphasized confidence in becoming an engineer; career planning and interviewing for jobs; understanding the wisdom of teaming; oral and written communication skills; industrial needs, requirements, and practices; and lifelong learning. Students enjoyed tackling open-ended design problems and were extremely satisfied, albeit surprised in some cases, with the extent of their learning under this course model. Students often remarked that this was one of the most time-consuming and challenging courses ever taken and, at the same time, was one of the most exciting and rewarding.

Employers, particularly the Employer Stakeholder Focus Group and external advisory boards, have also provided positive feedback. For example, on the subject of student frustrations with time demands of reporting and teaming, one employer said that if a student isn't spending 90% of his/her time writing or communicating with others, he/she is not communicating enough. Others have commented that the teaming and open-ended design experiences are very valuable and realistic. In addition, employers have found the course web site to be informative and useful.

6 Conclusions

In order to address the demands posed by the increasing role that embedded computing systems are playing within our society, the VESL project at Michigan State University is developing a multi-pronged approach for integrating concepts associated with embedded computing systems into the curricula. The approach incorporates embedded computing concepts from the high level software specification and development phases, to the real-time scheduler of the operating system, and further to the low level design of computer-based systems and digital circuits. Innovative approaches involving contemporary tools from

research and industry are used in laboratories to provide hands-on experience. Furthermore, innovative approaches to teaming are being used successfully to transfer important skills among students.

We plan to continue our development of course modules and laboratory assignments to support instruction in embedded computing. A major emphasis in the future will be to access instructional and laboratory components via the World Wide Web, to more easily allow students and universities to remotely use embedded computing systems laboratory equipment, software, and instructional modules located at Michigan State University, and to strengthen the collaboration between the University and industrial organizations working in the domain of embedded systems. An implementation and evaluation of this thrust will be performed with our Michigan project partners at Saginaw Valley State University and Lake Superior State University.

References

[1] M. Mutka, D. Rover, C. Wey, and B. Cheng, "Visions for embedded systems laboratories." Michigan State University, web. NSF Combined Research-Curriculum Development Program, URL:http://www.egr.msu.edu/VESL.

[2] Accreditation Board for Engineering Technology, "Abet criteria 2000," August 1995. Draft #4.

[3] L. Geppert, "Educating the renaissance engineer," *IEEE* Spectrum, September 1995.

[4] A. Speicher, "Asee project report: Engineering education for a changing world," in *ASEE PRISM,* December 1994.

[5] Computer Engineering Task Force, "A proposal for the computer engineering program in the college of engineering," July 1995. Michigan State University.

[6] P. Fisher, "Employer stakeholder focus group meeting." Minutes, June 28 1996.

[7] D. L. Parnas, "Education for computing professionals.," *IEEE Computer,* January 1990.

[8] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, *Object-Oriented Modeling* and Design. Englewood Cliffs, New Jersey: Prentice Hall, 1991.

[9] G. V. Coombs and B. H. C. Cheng, "Translation of object-oriented designs into vhdl," Technical Report MSU-CPS-98-8, Michigan State University, Department of Computer Science, 3115 Engineering Bldg., East Lansing, Michigan 48824, February 1998. (submitted for publication).

[10] R. H. Bourdeau and B. H. C. Cheng, "A formal semantics of object models," *IEEE Trans. on Software Engineering,* vol. 21, pp. 799-821, October 1995.

[ll] E. Y. Wang, H. A. Richter, and B. H. C. Cheng, "Formalizing and integrating the dynamic model within OMT," in ***Proc.*** *of IEEE International Conference* on *Software Engineering (ICSE97),* (Boston, MA), May 1997.

[12] E. Y. Wang and B. H. C. Cheng, "Formalizing and integrating the functional model into object-oriented design," in *(to appear in) Proc. of International Conference on Software* Engineering *and Knowledge Engineering,* June 1998.

[13] M. Mutka and J.-P. Li, "A tool for allocating periodic real-time tasks to a set of processors," *Journal of Systems and Software,* vol. 29, pp. 135-148, 1995.

[14] G. D. Micheli and R. Gupta, "Hardware/software co-design," in Proceedings *of the IEEE,* vol. 85, pp. 349-365, IEEE, March 1997.

[15] "EE 482 homepage." Michigan State University, web, 1997-98. URL:http://www.egr.msu.edu/classes/ee482/.

[16] D. Rover and P. D. Fisher, "Cross-functional teaming in a capstone engineering design course," in *Proceedings of the 1997 IEEE/ASEE Frontiers* in *Education Conference,* IEEE/ASEE, November 1997.

[17] J. Byrd and J. Hudgins, "Teaming in the design laboratory," *ASEE Journal of* Engineering *Education,* vol. 84, pp. 335-341, October 1995.

[18] K. Smith and A. Waller, *New Paradigms* **for** *College Teaching,* ch. Cooperative Learning for New College Teachers, pp. 185-209. Edina, Minnesota: Interaction Book Company, 1997.

[19] E. Aronson and et al, *The Jigsaw Classroom. Sage,* 1978.

[20] E. B. Nuhfer, New Paradigms *for College Teaching,* ch. Student Management Teams-The Heretic's Path to Teaching Success, pp. 103-126. Edina, Minnesota: Interaction Book Company, 1997.

[21] E. Mazur, *Peer* Instruction: *A User's Manual.* Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1997.

[22] J. Katzenbach and D. Smith, *The Wisdom of* Teams. Harper Business School Press, 1993.

[23] M. Aldridge and P. Lewis, "Multi-disciplinary teams: How to assess and satisfy ABET criteria," in Symposium *on Best Assessment Processes* in *Engineering* Education, April 11-12 1997. http://www.eng.auburn.edu/center/twc.

[24] J. Ganssle, "A question of balance," in *Embedded System Programming,* 1996.

[25] B. H. C. Cheng, "CPS 470 homepage." Michigan State University, web, Fall 1997. URL:http://www.cps.msu.edu/~cps470/F97-index.html.

Biographical Information

BETTY H. C. CHENG is an Associate Professor in the Department of Computer Science at Michigan State University. She received her B.S. degree from Northwestern University in 1985. She received her M.S. and Ph.D. degrees in Computer Science from University of Illinois at Urbana-Champaign in 1987 and 1990, respectively. She was a faculty fellow at the NASA Jet Propulsion Laboratory at the California Institute of Technology, where she investigated the application of formal methods to the space shuttle software. Her research and teaching interests include formal methods applied to software engineering, embedded systems, and object-oriented development.

DIANE T. ROVER is an Associate Professor in the Department of Electrical Engineering at Michigan State University and Director of the Computer Engineering Program. She received the B.S. degree in computer science in 1984, the M.S. degree in computer engineering in 1986, and the Ph.D. degree in computer engineering in 1989, all from Iowa State University. Her research and teaching interests include performance monitoring, evaluation, and visualization, parallel and distributing computing, embedded systems, and digital logic design.

MATT W. MUTKA is an Associate Professor in the Department of Computer Science at Michigan State University. He received the B.S. degree in electrical engineering from the University of Missouri-Rolla in 1979, the M.S. degree in electrical engineering from Stanford University in 1980, and the Ph.D. degree in Computer Science from the University of Wisconsin-Madison in 1988. He was a member of technical staff for Bell Laboratories from 1979-1982 and a visiting scholar at the University of Helsinki, Finland in 1988-1989. His research and teaching interests include computing systems and networks applied to multimedia networks, embedded systems, and real-time systems.