Evaluating the Impact of Real-time Systems Theory Course on a Multidisciplinary Embedded Systems Curriculum*

G. Singh[1], M. Mizuno[1], M. Neilsen[1], D. Lenhert[2], N. Zhang[3], A. Gross[4]

[1] Department of Computing and Information Sciences, Kansas State University (KSU), {singh,masaaki,neilsen}@cis.ksu.edu
[2] Department of Electrical and Computer Engineering, KSU, lenhert@ksu.edu
[3] Department of Biological and Agricultural Engineering, KSU, zhangn@ksu.edu
[4] The IDEA Center, 211 S. Seth Child Road, Manhattan, KS, agross@ksu.edu

Abstract

The ChERT project is focusing on developing an embedded system curriculum with the following innovative features: (a) an interdisciplinary curriculum and (b) integration of research results into teaching. The design of embedded systems requires knowledge from many different disciplines. For this purpose, we have designed a sequence of interdisciplinary courses for imparting embedded systems design education. The course entitled "Real-time Systems Theory" is a part of this curriculum. This course directly imports ideas developed as part of our research in embedded systems. The aim of this course is to provide students with a strong theoretical foundation for designing and analyzing embedded systems. This paper will present the motivation for designing this course, the importance of material covered in the course, and the impact of this course on the overall curriculum. We will report our experience in teaching this course and the feedback obtained from the students. The impact of having students from many engineering disciplines taking this course will be discussed.

1 Introduction

The number of embedded electronic systems used in automobiles, tractors, and other control systems continues to increase dramatically. Microprocessors currently used in embedded systems are as powerful as the large processors used only a few years ago. Development systems have also improved dramatically in the last decade, and use of high level languages in implementing

* This work was supported in part by NSF grant 9980321 and DARPA PCES program.

Copyright © 2002, American Society for Engineering Education

embedded systems is now common. With these advances, many complex features can be incorporated in embedded systems such as multi-tasking/threading and sophisticated network protocols that were only found in large operating systems several years ago. There exists a sufficient body of research in a number of areas such as real-time computing, fault-tolerance and networking that can be utilized to incorporate complex features in embedded systems. To train engineers to design and develop such systems, *these research results have to be integrated into a curriculum*.

Design and implementation of embedded systems requires a *broad knowledge in areas traditionally not covered in any one discipline*. These areas include (a) Computer Science (which deals mainly with high-level software), (b) Electrical and Computer Engineering (which deals mainly with hardware and low-level software), and (c) other engineering disciplines, such as Agricultural and Biosystems Engineering (which deals with application development and numerous other peripherals such as sensors and actuators) and Mechanical and Nuclear Engineering (which deals with control automation for safety critical systems). As a result, it is very difficult to train students and engineers who can effectively design and implement embedded systems.

The ChERT (researCH in Embedded Real Time systems) project at Kansas State Univeristy is funded the NSF Combined Research and Curriculum Development program and is focusing on developing an embedded system curriculum with the following innovative features: (a) *an interdisciplinary curriculum* and (b) *integration of research results into teaching*. The curriculum consists of a sequence of four courses. The first course is a remedial course consisting of three modules (data structures, concurrent programming, and real-time electronics) to bring students from different backgrounds up to speed. The second course again consists of three modules, namely real-time programming fundamentals, real-time operating systems and real-time electronics. This course is required by all students in the curriculum and introduces students to concepts that allow simple but complete embedded systems to be developed. The fourth course is the Capstone design course in which a major industrial-sized embedded system is designed by students.

The third course in the sequence, Real-time Systems Theory, is intended to teach techniques for design and analysis of an embedded system. Typically, an industrial-sized embedded system involves a large number of components interacting with one another in complex ways. Although the second course teaches principles to design simple embedded systems, it is not sufficient to deal with the complexity of larger embedded systems. In the third course, we familiarize students with object oriented design techniques, aspect-oriented programming approach based on Unified Modeling Language of Rational Unified Process (which is an industry standard), schedulability analysis and verification. Many of these concepts have been developed as part of our embedded systems research. The topics covered in this course provide knowledge of various design and analysis techniques to manage the complexity of large embedded systems. This course is an integral part of curriculum as it allows students to make the jump from concepts learned in the second course to design industrial-sized systems in the Capstone Design course.

Copyright © 2002, American Society for Engineering Education

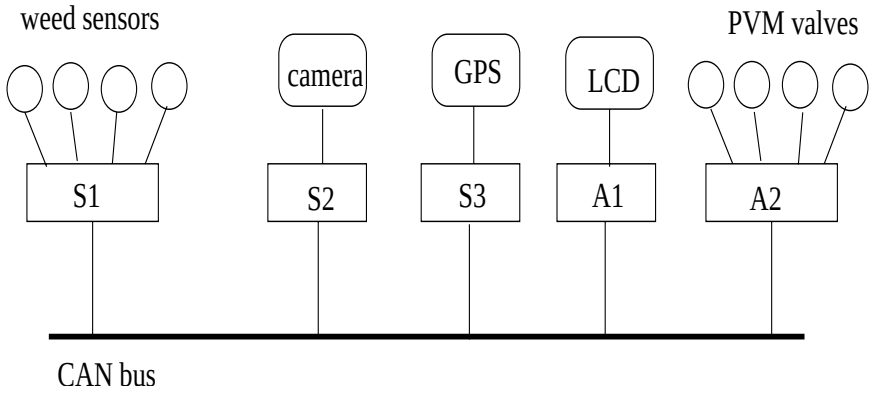

Figure 1: Architecture of the PA application.

This paper is organized as follows. We first present the basic structure of an embedded system and discuss its characteristics that influenced our design methodology and contents of the course. Subsequently, we discuss the concepts covered in the course. In Section 4, we discuss the overall embedded systems curriculum and the impact of the Real-time Theory course on the curriculum. In Section 5, we present the student evaluation, feedback obtained, lessons learned and future directions for this course. Finally, we present our conclusions in Section 6.

2 Motivation for the Real-Time Systems Theory course

A typical embedded system may consist of a large number of devices (sensors and actuators) and separate processors that control the devices which often results in a large number of complex, interacting activities. Figure 1 shows the architecture of a Precision Agriculture (PA) application for customizing the rates and/or compositions of farm inputs such as herbicides, pesticides, and fertilizers at every location in the agricultural field. The example system consists of various sensing and actuating devices controlled by Infineon C167CR microprocessor based controller modules. C167CR controller modules are equipped with various on-chip interface modules including serial-interface module, analog input module, digital interface module, and pulse interface module. All controller modules are connected via a Controller Area Network (CAN). A prototype of this system has already been developed [WZO+00]. In the following, we list several characteristics of an embedded system that influenced our design methodology and the course contents.

- Embedded systems often have a large number of components that may interact with one another in complex and sometimes unpredictable ways. Such interactions may involve *execution dependencies* (for example, in Figure 1, the optical weed sensor must compute the weed density for each GPS reading sent by the GPS unit). In on-going research, we are working on more complex applications (with 10^3 components) designed using Boeing's BoldStroke system [Sha99] which, in addition, involve *data dependency* interactions due to shared data.

Copyright © 2002, American Society for Engineering Education

- Embedded systems are often dynamic in nature wherein new components may need to be plugged in and older components may have to be removed. In addition, interaction between the components may have to be restructured ($e.g.,$ to further constrain the system to satisfy additional functional or performance requirements).
- Embedded systems are being increasingly used in safety-critical systems ($e.g.$, avionics, automotive and medical domains). For such applications, verification of embedded systems is essential. Traditional design techniques often rely on testing which may not be suitable for such domains.
- In addition to functional requirements, an embedded system may have several non-functional requirements such as real-time deadlines, synchronization and fault-tolerance. For example, schedulability analysis of a system may be needed to ensure the real-time deadlines are satisfied.

As illustrated by these requirements, the complexity of embedded systems demand that embedded systems engineers be provided with powerful design and analysis techniques. The design of embedded systems must be modular, and component based (so that they can be restructured if necessary), and embedded system engineers must have knowledge of analysis and verification techniques. Object-oriented analysis and design methodologies have been very successful in creating highly reusable software systems. In particular, the Rational Unified Process (RUP) in the Unified Modeling Language (UML) has become the *de facto* standard in software development [JBR99]. When applying RUP in the development of embedded systems, its *use-case driven* methodology effectively produces a set of sequential programs, each of which controls a specific type of devices. In embedded systems, these programs are often required to interact with each other. More precisely, these programs are executed concurrently by multiple threads, and such threads synchronize with each other for collaboration. Thus, the designer is confronted with several issues such as those of synchronization, scheduling, and distribution, related to the interactions between the concurrent threads. The current RUP, however, does not provide a standard methodology to develop code for such interactions.

We have developed an aspect-oriented approach to develop embedded systems [MSN00] that involves two stages. In the first stage, the functional code is developed using RUP. The development of the functional code starts with the identification of *actors* who interact with the system by sending stimuli and receiving responses. Use cases are used to specify the way actors interact with the system. In the next step, we consider how the use-cases are realized by (1) identifying the necessary classes and the relationships between those classes (Class Model) and (2) describing use-case realizations which describe the collaborations between instances of the classes. Sequence diagrams, collaboration diagrams, and scenarios are often used to describe use-case realizations. Finally, in the implementation step, each use-case realization is implemented on a target architecture. The

Copyright © 2002, American Society of Engineering Education

second stage involves the development of aspect code for various properties such as synchronization, fault-tolerance and distribution. In this phase, various analyses such as those related to scheduling and verification are also carried out. We have incorporated results from this methodology into the real-time theory course which we describe next.

3 Real-time Systems Theory Course contents

The Real-time Embedded Systems Theory course is intended to teach techniques for design and analysis of an embedded system. The course directly imports ideas developed as part of our research in embedded systems. The aim of this course is to address the issues outlined in the previous section and provide students with a strong theoretical foundation in designing and analyzing embedded systems. This course is divided into three modules:

1. The first module deals with object oriented design techniques. As discussed above, this module introduces students to the Rational Unified Process model. The concepts of use cases, scenarios, and use case realization are introduced. Subsequently, we teach our aspect oriented methodology for developing embedded systems. A large example system (a telephone call distributor system) is used to illustrate the methodology.
2. The second module is concerned with analysis techniques. In particular, the focus is on real-time scheduling and schedulability analysis. Various classes of scheduling algorithms such as clock-driven scheduling, priority-driven scheduling, and priority-inheritance protocols are discussed. Scheduling in real-time networks such as controller area networks and its analysis is also discussed.
3. Verification and testing is the subject of the third module. The goal of this module is to familiarize students with verification and testing techniques and introduce them to appropriate tools. In particular, the SPIN tool and its specification language Promela is used [Hol91]. This tool has a widely used model checker and has been extended to deal with real-time properties. Various example systems are used to illustrate common types of errors in embedded system programming such as concurrent access in a multiple ATM system, and race conditions in electronic circuits.

The duration of each module is five weeks. In this course, we assume that students are proficient in programming but may not have exposure to modular design techniques, analysis and verification. This course was taught in Spring 2001.

Copyright © 2002, American Society for Engineering Education

4 Overall Embedded Systems Curriculum

In this section, we describe the overall embedded systems curriculum. Subsequently, we discuss the relationship and importance of the theory course to the curriculum.

- **Remedial Course**: This course consists of three modules that cover basic electronics, data structures, and concurrent programming respectively. They are intended to fill deficiencies for students from different backgrounds (such as engineering departments other than CS, CE and EE).
- **Implementation Course**: This is the first core course that allows students to work with state-of-the-art design tools, embedded development environments, and target platforms to interconnect a variety of sensors and actuators in complete, but simple real-time embedded systems. Topics covered include the relationship between C/C++ constructs and corresponding assembly code generated by compilers, real-time operating systems, real-time network protocols, and the interconnection of peripherals.
- **Real-Time Systems Theory**: This is the second core course that covers design and analysis of real-time embedded systems, including language and operating system support, scheduling, feasibility analysis, fault tolerance, object-oriented design methodology and tools, verification, and validation.
- **Capstone Design Course**: This is a project-based course to design and implement a complete and comprehensive real-time embedded system.

Since this is an multidisciplinary program, we have developed three complementary physical labs, with one in each participating department. We have also developed a prototype virtual lab to interactively use our embedded system lab resources over the web. Initial testing conducted to date with the virtual lab indicates that network bandwidth requirements are quite manageable, even for students logging in from home on a 56K line.

Relationship of the Theory course to the curriculum:
The second course in the curriculum teaches principles to design simple embedded systems. For example, in Fall 2001, students designed a controller (for acceleration, braking and steering) for a car with stepper and DC motors. However, the concepts taught in this course are not sufficient to deal with the complexity of larger embedded systems. An industrial-sized embedded system may involve a large number of components interacting with one another in complex ways. Hence, significant emphasis must be placed on software engineering principles during the design stage. In addition, currently most embedded systems code is developed using a high-level language, only a small part is typically written in assembly language. Therefore, students must learn the design tools and techniques available for object oriented in high-level languages. Furthermore, the embedded systems designer must also have knowledge of analysis techniques to ensure that real-time

Copyright © 2002, American Society for Engineering Education

deadlines are met and to provide assurance that the system will satisfy its required functionality. The intent of the theory course is to familiarize students with these concepts. As discussed in Section 3, we teach students object oriented design techniques and an aspect-oriented approach based on UML. We also teach techniques for analyzing embedded systems such as schedulability analysis and verification. We believe that this course is an integral part of curriculum that allows students to make the jump from concepts learned in the second course to design industrial-sized systems in the Capstone Design course.

5 Course Evaluation and Assessment

• **Course Delivery Process**:
In Spring 2001, students were asked to evaluate the effectiveness of different course elements in terms of how they contributed to their learning. Various course elements such as lecture notes, assignments, and web resources were used to evaluate the course delivery process. As shown in the following table, the students were overwhelmingly positive about the course delivery process. Students used a 5-point scale (1 = Strongly disagree, 2 = Disagree, 3 = In between, 4 = Agree, 5 = Strongly agree). % positive reflects ratings of 4 or 5; % negative reflects ratings of 1 or 2. Percentages may not be equal to 100 because those responding "in-between" and those omitting the items are not reported.

	Lectures	Handouts	Assignments	Web Resources
% positive	83	86	80	65
% negative	3	3	4	5

• **Amount of Student Learning**:
The amount of student learning was also evaluated. To evaluate prior knowledge, students were asked to evaluate the statement *"Before I took this course, I knew a lot about ...[specific topic]....* The *specific topics* listed were related to concurrent programming, verification, specification of properties, object oriented techniques, scheduling, and "synchronization". As seen from the following table, most students had some prior knowledge but not a great deal (only 16% indicated that they had prior knowledge). To assess learning, we also asked to evaluate the statement *"I learned a lot about ...[specific topic]...* for each topic. Students were generally positive about the amount of material learned.

	Prior knowledge	Amount of learning	Overall learning
% positive	16	76	78
% negative	53	6	3

Copyright © 2002, American Society of Engineering Education

• **Impact of students from different disciplines**:
One of the challenges we faced in this course was that students from many different disciplines were enrolled in the course. Although the first two courses in the sequence did address this problem to some extent by bringing students deficient in some topics up to speed, we found that students still needed more time to grasp the topics covered in this course (the duration of each module in the course is 5 weeks). In our survey, the students were asked to respond to the question *Which content areas or topics should be covered in more detail or given more time*. In response to this statement, the students listed a number of topics from each of the modules. When asked *Which content areas or topics should be covered in less detail or given less time*, very few topics were listed. From the survey, we also found that students did understand the relevance of the material covered in this course to designing embedded systems, but they were concerned that too many topics were being covered in one course. Specifically, we feel that for students from disciplines other than computer science, many of these concepts are new. As a result of this feedback, we are in the process of restructuring the course sequence to allow *multiple entry points* based on the skills and needs of the users. We will target the following three types of users:
(a) Those who have limited software engineering background: A typical profile of such users is those who only need to reconfigure existing embedded systems by adding/removing/exchanging some devices. For such users, we intend to develop a plug-and-play approach whereby they can select modules from a library, link them, and download the resulting software to the target platform.
(b) Students and professionals whose main interests are in development of mechatronics devices and their applications in embedded systems: A typical profile of these users is those who need to develop embedded systems with various devices to fit their needs. Their backgroond includes skills to develop sequential programs in high-level languages to control their devices. They need to be taught a pattern-based approach to develop aspect code such as synchronization, real-time, and fault tolerance aspects.
(c) Students and professionals with strong mathematics and computer science and/or engineering backgrounds. Such students need to be taught the methodology and theoretical foundations in detail so that they can develop code for any type of embedded system application and target platform.

6 Conclusions

With the proliferation of electronic components on board modern equipment, automobiles, appliances and other control systems, there is an increased demand for professionals trained to develop embedded electronic systems. We are developing a multidisciplinary curriculum for training students to design embedded systems. Due to the complex nature of embedded systems, appropriate design and analysis tools are required to manage the complexity of embedded systems. The

Copyright © 2002, American Society of Engineering Education

Real-time Systems Theory course provides students with the foundation require to use appropriate design and analysis tools. This paper presented the motivation for designing this course, its course contexts and its impact on the overall curriculum. We also presented the student evaluations for the course delivery process and the amount of learning.

References

[Hol91] G. Holzmann. *Design and Validation of Computer Protocols*. Prentice Hall, 1991.

[JBR99] I. Jacobson, G. Booch, and J. Rumbaugh. *The Unified Software Development Process*. Addison Wesley, 1999.

[MSN00] M. Mizuno, G. Singh, and M. Neilsen. A structured approach to develop concurrent programs in uml. In *Proceedings of the Third International Conference on the Unified Modeling Language*, 2000.

[Sha99] D. Sharp. Avionics product line software architecture flow policies. In *Proceedings of the Digital Avionics Systems Conference*, 1999.

[WZO+00] J. Wei, N. Zhang, N. Wang D. Oard, Q. Stall, D. Lenhert, M. Neilsen, M. Mizuno, and G. Singh. Design of an embedded weed-control system using controller area network (CAN). In *Proc. of ASAE*, 2000.

Biographical Information

GURDIP SINGH is an Associate Professor in the Department of Computing and Information Sciences at Kansas State University. His research interests include networking, distributed computing, design techniques and verification.

MASAAKI MIZUNO is a Professor in the Department of Computing and Information Sciences at Kansas State University. His research interests include operating systems and distributed systems.

MITCH NEILSEN is an Assistant Professor in the Department of Computing and Information Sciences at Kansas State University. His research interests include real-time embedded systems, distributed systems and distributed scientific computing.

DONALD LENHERT is the Paslay Professor in the Department of Electrical and Computer Engineering at Kansas State University. His research interests include real-time embedded systems and digital testing.

NAIQIAN ZHANG is a Professor in the Department of Biological and Agricultural Engineering at Kansas State University. His research interests include sensors and control for biological and agricultural systems.

AMY GROSS, Associate Director of the IDEA Center, served as an external evaluator for the NSF CRCD grant. The IDEA Center's mission is to assist colleges and universities access and improve teaching, learning and administrative performance.

Copyright © 2002, American Society of Engineering Education