Real-Time Computing in Software Engineering Education

(CSEE&T 2000 Workshop Position Paper)

Andrew J. Kornecki
Department of Computing and Mathematics
Embry Riddle Aeronautical University
Daytona Beach, FL 32114
e-mail: korn@db.erau.edu
WWW: http://faculty.erau.edu/korn

An incredible growth of the computing power, advances in microelectronics, telecommunication, and new software tools and techniques do allow us to have computers controlling, displaying, supporting, even, as some would like to have, thinking for us. In addition to a well established real time applications in military, aerospace, aviation, and medical systems, nowadays the consumer electronics is an area that uses more and more systems with real-time features. For all these systems time criticality and determinism are important, but often safety and reliability are of an equal importance. The software development of such systems requires skills and knowledge exceeding the standards offered by colleges and universities in most computer science and engineering programs. Real-time instruction in majority of electrical and control engineering program focus on hardware and control algorithms. Computer science and software engineering programs tend to focus on issues of software design and theoretical aspects of operating systems schedulability. It is not so often, that the students get proper background to develop the target system application software with full understanding of the hardware and operating system implications. Future software developers must understand the basic concepts distinguishing real-time applications from standard non-real-time applications. The industry needs engineers with knowledge of specialized time-critical reactive systems. Concepts developing software in a host-target environment, software timing, multitasking, intertask synchronization and communication, resource contention, external devices and interrupt handling need to be taught. Graduates who understand how the application software interacts with the operating system and the environment are in high demand.

University laboratories equipped with a general purpose operating systems (Windows or one of the flavors of UNIX) limit the software development to data processing programs with standard input/output. In an effort to provide students with such knowledge, the Embry-Riddle Aeronautical University has been offering real-time courses. We designed a dedicated real-time laboratory, with a support from the National Science Foundation, and affiliated real-time software vendors. The laboratory contains computers and interface devices for teaching real-time concepts. Resources include a wide variety of hardware and software platforms supporting experiments with and development of real-time software.

Several approaches have been tested over the course of the last five years. We did offer undergraduate and graduate courses focusing on requirements and design issues with the implementation of a soft real-time using multitasking Ada on UNIX platforms. We designed series of C/POSIX experiments with concurrency implemented in a form of UNIX processes on various platforms (RS6000/PowerPC of IBM, Indigo/O2 of Silicon Graphics, Sparc10 of Sun Microsystems) using System V standard. We introduced multithreading with POSIX thread libraries and experimented with implementation on Intel-based native real-time operating systems (QNX, LynxOS). We developed project on VME Motorola 68K boards running VxWorks with Sun hosted Tornado1 development environment. In the recent offering

we are using Tornado2 Integrated Development Environment hosted on Window NT platforms.

The courses have their project component, where student teams work on implementation of real-time software delivering the software cycle artifacts including prototype demonstration, formal presentation and Web-based report. In the early part of the course students go through series of laboratory assignments experimenting with software implementation on selected platforms while mastering the basic real-time concepts taught in the class. The experiment sequence was designed taking into consideration availability of platforms and tools. We have experiments running on System V easily portable to any lab with UNIX workstations. We developed experiments for both C and Ada implementations. We have experiments using POSIX threads library. And finally we have extensive experiments based on host-target environment with VxWorks. The second consideration was availability of the experiment documentation. Posting the material on the Internet, accessible by the students from their course web pages, was the natural solution. The ability to cross-link documents and providing example source code on-line is a great help in mastering the course material. As our e-mail shows, the web materials have been often used for training by industry employees and students from other universities.

It appears that the students want to learn the implementation specifics and are eager to learn about application software interactions with the external devices via operating systems primitives. They enjoy the experiments in the lab, particularly when the final outcome is a system moving industrial robot, simulation of a space station alarm facility, aircraft traffic collision avoidance system, or a control of lunar explorer. The student projects are following the software lifecycle with focus on implementation. These senior level undergraduate and graduate courses are often feared as being too demanding and time consuming. However, months later from their industrial job site, the alumni send to the instructor e-mails praising the approach taken.

The workshop shall discuss briefly issues of Real-Time Specification and Design while focusing on characteristic features of real-time systems, role of RT in software engineering education, selected notations for real time specification and design, CASE tool features and selection, and real-time software design methodologies. Some comments on Real Time Programming Concepts include programming language selection, access to time, concurrency (multitasking/multithreading), the synchronization and communication issues, interrupts and exception handling. The Real-Time Operating Systems component shall address issues of responsiveness and timeliness, schedulability and predictability, preemption and priority-based scheduling, and methodologies of real time operating system selection.

In addition to the author, the workshop participants are:

- Janusz Zalewski of the University of Central Florida, Orlando, FL, who shall present a statement on Automatic Development Tools in Software Engineering Courses.
- David A. Dampier, Mississippi State University, Mississippi State, MS, and Ronald E. Wilson, INDUS Corporation, Vienna, VA, who shall present a statement on Teaching Scientific Method for Real-Time Software Engineering.

Two other participants expressed interest in sharing their experience on teaching real-time concepts in software engineering curricula. They are: All Mok, Dept. of Computer Science, University of Texas, Austin, TX, and Sadegh Davari, Software Engineering Program, University of Houston, Houston, TX.