World Transactions on Engineering and Technology Education
Vol.2, No.1, 2003

The development of an environment for remote embedded systems: feedback from students and subsequent enhancements

Bruce D. Moulton, Vladimir L. Lasky & Stephen J. Murray

University of Technology Sydney
Sydney, Australia

ABSTRACT: A subject called *Operating Systems* is taught to about 400 students at the University of Technology Sydney, Sydney, Australia, each year. This subject covers rapidly changing domains including software, hardware, applications and embedded systems. Until 2001, students of the subject learnt much of the material by undertaking tasks in an operating systems laboratory. In 2001, a new approach was developed that enabled students to complete embedded systems programming tasks from outside the laboratory. This approach includes a remotely accessible embedded system development laboratory with Web-based vision and support. The new environment provides users with the ability to remotely control, monitor and program real embedded system components. This article describes the remote development environment, the feedback from students, as well as subsequent enhancements.

INTRODUCTION

Trends in industry influence how we perform practice-based education. Remote development is a process where engineers in different geographic locations participate in projects. The process has enabled vast amounts of open source software to be developed. Currently, there is an increased demand for engineering graduates who can perform competently in remote development environments.

In the higher education sector, there has been increasing emphasis on flexible learning, including online learning. This has presented new opportunities as well as challenges. At an increasing number of universities, Web-based courseware is used in teaching laboratory-based courses [1]. Such developments have led to concerns about the implications of decreased face-to-face teaching [2][3].

An undergraduate engineering course at the University of Technology Sydney (UTS), Sydney, Australia, has a practice-based educational framework. The course encourages students to become reflective engineering practitioners who can apply their knowledge and skills in a technically competent and ethical manner.

The Faculty of Engineering at the UTS has attracted increasing numbers of students in computer systems engineering and software engineering. Enrolments in these areas have increased at such a rate that the Faculty has needed to consider alternatives to traditional campus-based laboratory work in some subjects, including the subject called *Operating Systems*.

In 1999, the Faculty joined a national consortium concerned with the development of embedded systems. In 2001, *Operating Systems* was modified so that the laboratory work would focus on embedded systems development.

The UTS Quality Development Unit conducts Student Satisfaction Surveys every two years. In 2001, a representative sample of 3,000 enrolled students received a survey questionnaire. The response rate was 33%.

Seven key improvement priorities were identified, namely:

- The provision of timely and constructive feedback on learning.
- Access to computers.
- The provision of useful and relevant learning materials and equipment.
- Relevant assessment tasks.
- Ensuring close links between theory and practice.
- Development of the ability to solve practical problems successfully.
- Clear assessment requirements [4].

An embedded systems development environment has been developed in accordance with these priorities.

WHAT ARE EMBEDDED SYSTEMS?

Embedded systems are specialised computer systems that are part of a larger system or machine. Many examples are found in consumer electronics – virtually all appliances that have a digital interface have embedded systems. Examples of these include microwave ovens and videocassette recorders.

An embedded system typically consists of a microcontroller, some interface circuitry and a program stored in non-volatile memory. More sophisticated embedded systems typically run their programs under an operating system to provide support for concurrency and to make use of system calls. Simpler systems are typically standalone and do not use an operating system.

DESCRIPTION OF THE ENVIRONMENT

Students of *Operating Systems* now make extensive use of remotely accessible embedded system hardware and software, as well as a Web-based peer-supported learning environment. This environment enables students to remotely control, monitor and program real embedded system components [5].

The learning environment has three functionally distinct parts, namely:

- A Web-based peer-supported learning environment.
- Web-based low-latency video monitoring.
- Remotely accessible embedded system hardware and software.

The first part of the environment is known at UTS as *UTSOnline*. This is a well-established teaching and learning tool that is used in many of the University's subjects. Teachers and learners use *UTSOnline* to carry out tasks such as distributing and accessing course materials, assignments and conducting online discussions.

The second and third parts of the environment are specific to the subject *Operating Systems*. They have been developed so that students can complete most of their embedded systems tasks from outside the embedded systems laboratory.

The system, depicted diagrammatically in Figure 1, provides students with the tools to undertake real embedded system projects.

An advantage of the approach is that laboratory access is not substituted by simulation. The system makes it possible for UTS staff and students to remotely develop embedded system components that may have research applicability or commercial value.

SERVER SIDE HARDWARE

The development platform consists of development boards, peripheral boards, servers and cameras. The environment has a clustered configuration. The configuration allows individual boards, servers and cameras to be temporarily taken offline for maintenance.

A master server holds user accounts and development utilities. It runs Linux and includes a RAID5 (Redundant Array of Independent Disks) providing 80Gb of storage. This server is connected to the Internet via an Ethernet connection, and to three board servers.

Each of the three board servers is connected to the master server and to four development boards via communications and debugging ports.

The development boards are single-board computers based on the Motorola C*oldfire* processor, a chip widely used in devices such as routers and portable MP3 players. Each board includes an MCF5206e processor running at 54 MHz, 16MB of DRAM, 1MB Flash ROM, two RS-232 serial ports, Background Debugging Mode (BDM) support (which allows remote on-chip debugging) and a number of other input/output components [6].

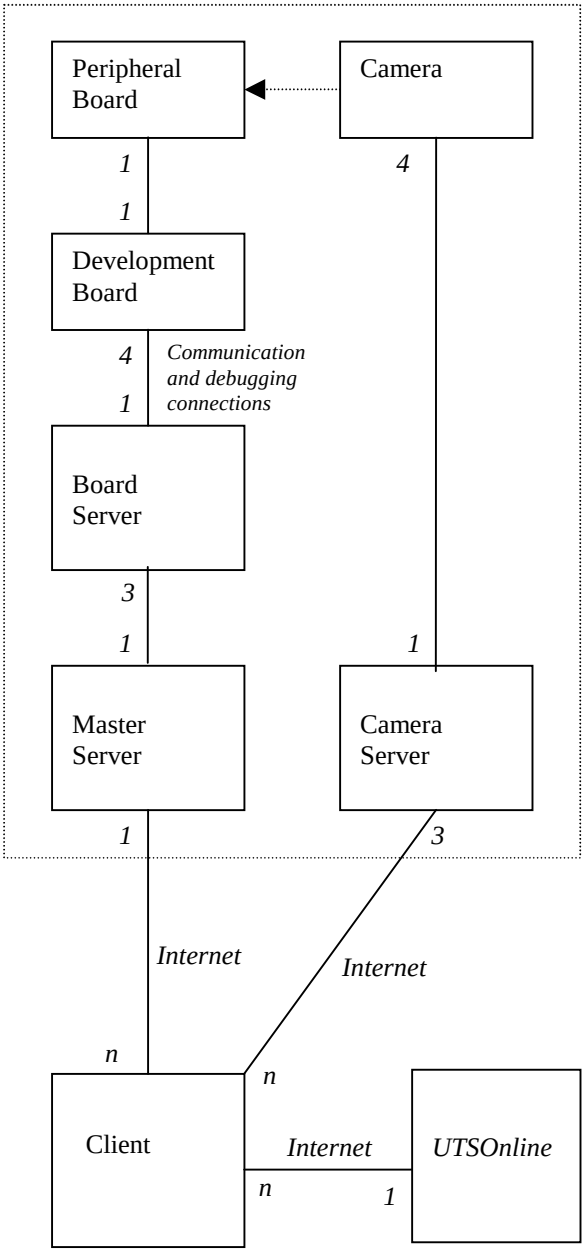

Figure 1: The environment.

The peripheral boards are attached to the development boards and have seven segment displays that can be controlled by applications running on the development boards.

The camera servers are connected to cameras that provide images of the peripheral boards. These provide remotely located students with live MJPEG/MPEG4 images. It is possible to control what is displayed on a peripheral board and see it using a Web browser.

SERVER SIDE SOFTWARE

Software at the server side includes:

- *GCC-m68k*: the GNU cross-compiler configured for Motorola 68k series processors and derivatives (this includes the *Coldfire*). There are two versions currently available: one that produces standalone programs, as well as another that produces programs in μCLinux (a derivative of Linux developed specifically for microcontrollers).

- *GDB/BDM*: a customised version of the GNU debugger with support for remote debugging via the BDM interface. This allows students to perform real-time hardware debugging of programs while they are running on the development board.
- So-called *arbitration* software that provides a mechanism for fairly allocating development-board connections to the incoming requests. The software maintains a queue of students wishing to initiate connections and enforces time limits on these connections. It stores resources in a PostgreSQL database, and records statistics such as average usage per session and hourly usage. This software was developed in-house.

CLIENT SIDE SOFTWARE

Software installed on students' desktop machines for accessing the server (and thereby a development board) includes SSH and SCP clients. SSH (Secure SHell) is used in place of *telnet* for accessing the server, while SCP (Secure Copy) is used in place of FTP for transferring files between the students' computers and the servers.

EXERCISES THAT USE THE ENVIRONMENT

Students use the new environment to remotely run and debug programs on the boards. This enables students to complete most of their embedded system projects from off-campus (or from general purpose computer laboratories). Students use the environment to complete projects where they:

- Develop POSIX (Portable Operating System Interface) applications running on μClinux;
- Port applications from their desktop to an embedded Linux environment;
- Develop μCLinux device drivers.

In completing these assignments, students become familiar with critical aspects of embedded systems development, such as kernel environment. Students make extensive use of online communication and self-help facilities.

FEEDBACK

Students gave feedback via independently administered Subject Feedback Surveys. These surveys asked students to rate the following statements:

1. This subject was relevant to me.
2. The subject was delivered in a way that was consistent with its stated objectives.
3. I had a clear idea of what was expected of me in this subject.
4. My learning experiences in this subject were interesting and thought provoking.
5. I found the assessment fair and reasonable.
6. There were appropriate resources available to support the subject.
7. I received constructive feedback when needed.
8. Overall, I am satisfied with the quality of this subject.
9. What did you particularly like in this subject?
10. Please suggest any improvements that could be made to this subject [7][8].

Students of *Operating Systems* were surveyed in both the autumn and spring semesters of 2002. In both semesters, students presented mixed responses to the system.

Some students reported having difficulty relating to the system they were using. In some cases, these difficulties were attributed to the system being physically remote, making it difficult to visualise what it was they were actually doing with the system.

In both of the semesters completed so far, two of the most frequent concerns expressed by students resulted from there being a limited number of the microcontroller boards. These were:

- Students reported that they did not like waiting in the (online) queue in order to gain access to the boards.
- Students reported that they felt pressured by the time limits imposed on them, particularly when attempting to solve complex programming problems.

Many students appreciated the practical nature of the exercises they had been able to complete. Some students stated that the exercises had industrial and commercial relevance.

Improvements

The performance of the system has since been improved by several measures including the addition of the master server and the phased augmentation and subsequent replacement of the microcontroller boards, increasing their number from four to 12.

Maximum waiting times have so far been reduced from 4.5 hours to 10 minutes. Once allocated a board, students are now able to remain connected to it for one hour (previously 30 minutes).

CONCLUSIONS

Feedback indicated that previous limitations of the system affected its usability. Subsequent enhancements have improved the performance of the system and should thus improve its usability.

REFERENCES

1. Gomes, V.G., Choy, B., Barton, G.W. and Romagnoli, J.A., Web-based courseware in teaching laboratory-based courses. *Global J. of Engng. Educ.*, 4, **1**, 65-71 (2000).
2. Edwards, S., Can quality graduate software engineering courses really be delivered asynchronously on-line? *Proc. Inter. Conf. on Software Engng.*, Limerick, Ireland, 676-679 (2000).
3. Lilja, D.J., Teaching computer systems performance analysis. *IEEE Trans. on Educ.*, 44, **1**, 35-40 (2001).
4. UTS Students Association and the Office of the Pro-Vice-Chancellor Education and Quality Enhancement, UTS Report on the 2001 Student Satisfaction Survey. Sydney: UTS (2001).
5. Moulton, B.D., Lasky, V.L. and Carmody, N.J., Practice-based engineering education: a distributed environment for teaching and learning embedded systems *Proc. 5th UICEE*

Annual Conf. on Engng. Educ., Chennai, India, 33-35 (2002).
6. Payne, G., The Coldfire Project. Unpublished UTS internal report (2002).
7. The University of Technology Sydney (UTS) Quality Development Unit, Subject Feedback Survey - Autumn 2002. Sydney: UTS (2002).
8. The University of Technology Sydney (UTS) Quality Development Unit, Subject Feedback Survey - Spring 2002. Sydney: UTS (2002).