Curriculum Development of Hardware Interfacing System for Visual C++ in Computer Engineering Course

Martinez, D.T. [1]
Lecturer
University of Auckland
Auckland, New Zealand

Taylor, H.S
Associate Professor
Victoria University of Wellington
Wellington, New Zealand

Ganiron Jr, T.U
Associate professor
AUT
Auckland, New Zealand

Conference Topic: Curriculum Development, Continuing Engineering Education and Lifelong learning

INTRODUCTION

The University of Auckland known to be a caring and committed university, has for its vision the development of professionals, leaders, and citizens towards the transformation of the City of Auckland and the nation. Its Computer Engineering Department is dedicated and committed to nurture and produce competent engineers. Academic excellence and technical competence are instilled in the minds of the University of Auckland engineering graduates. They are well grounded on technical aspects, and updated on the technological advances and the prevailing conditions [1,10].

The reason of the need to re-engineer the curriculum of hardware interfacing system for visual C++ is to keep its students abreast with expanding trends in technology in accordance to the ABET standard and to be globally competitive [12]. Focus and emphasis of the revision and areas of consideration done during the first periods in University of Auckland engineering academic history, it enables the department to establish a steady pattern and instructional tends which may serve as a basis for multidimensional formulations of changes in response to dynamic needs of the students and contemporaries.

1 COMPUTER ENGINEERING PHILOSOPHY

Undergraduate computer engineering education continued to change and, has for the most part, kept pace with the new topics in the field. However, the pedagogy has not changed significantly. At the University of Auckland, the curriculum effects what is a common approach to teaching students. Curricula emphasize [10]: (a) The

[1] Martinez, D.T.

construction of relatively small programs, at most a few hundred lines. (b) The use of a programming language that is used rarely outside of undergraduate courses. (c) The development of programs ‘‘afresh’’ for each assignment or course. (d) A development environment lacking modern tools. (e) Programming in isolation or in small groups at best. (f) The belief that if a program ‘‘works’’ it is acceptable. (g) An informal development approach rather than one that is rigorous and exercises analytic skills. Comparing the content of the curriculum with the situation the real world, the researcher sees a considerable contrast. Practicing computer scientists have to deal with the antithesis of what to teach: (a) Software systems that are often thousands or even millions of source lines long. (b) Tasks that involve modifying such systems rather than developing them. (c) Existing systems that might be very old but remain important and have to be maintained. (d) Tool-rich working environments. (e) Development efforts that are undertaken by large teams. (f) The realism of cost/performance trade-offs in business contexts. (g) System development according to mandated processes and standards. (h) Expenditure of considerable effort on tasks other than source-code development.

This list illustrates the general range of skills needed by a contemporary computing professional. This set of skills is the antithesis of what the teachers teaching. Clearly, there is a serious mismatch between what is taught, how it is taught, and the emphasis it receives on one hand, and what the consumers of the education actually need on the other. The new curriculum of Hardware Interfacing System for Visual C++ driven by the desire to make the necessary changes to accommodate the educational needs of the future computing professional. These changes necessitate a complete revision of the content, approach, and resources used in the undergraduate teaching program [10,11].

The incorporation of these ideas into teaching has leads to the development of an entirely new undergraduate curriculum. The Hardware Interfacing System for Visual C++ course of the new curriculum is designed to be more mathematically rigorous, more practice-oriented, more closely related to the real world environment. This has leads to a number of substantial effects on the program, specifically [1,2]: (a) Extensive revision of many existing courses. The Hardware Interfacing System for Visual C++ course containing elements of programming, data structures, machine representation, etc. have had to be revised extensively. (b) Development of several new courses, especially in discrete mathematics and computation theory. (c) The replacement of the programming language used for teaching purposes. C++ was selected as the new programming languages and, although not ideal, it has the benefits of supporting object-oriented programming and increasing industrial acceptance. (d) A belief in the importance of reuse of courseware. (e) The development of the “closed laboratory” facility to support closed-laboratory exercises. The facility is being expanded to include devices such as motorized vehicles and other elements that simulate realistic applications.

2 COURSE DESIGN

2.1 Hardware Interfacing System for Visual C++

The development of device drivers and embedded software is full of challenges. But it is possible to write easy-to-read portable C code to control peripherals ranging from simple timers and UARTs to complex custom FPGAs. This important course covers a large number of subjects ranging from C start up code and the “world before main()” to the development of interrupt-based device drivers that

interact with peripheral control and status registers through memory-mapped I/O, overlaid struts, and bit fields[3.4].

In general, topics covered can be classified under five areas, i.e., number representation, assembly language, basics of c, memory management, operating-system process model, high-level machine architecture, memory hierarchy and implementation of high-level languages.

This course examines key computational abstraction levels below modern high-level languages; number representation, assembly language, introduction to C, memory management, the operating-system process model, high-level machine architecture including the memory hierarchy, and how high-level languages are implemented [5.6]. The University of Auckland will develop students' sense of "what really happens" when software runs — and that this question can be answered at several levels of abstraction, including the hardware architecture level, the assembly level, the C programming level and the Java programming level. The core around which the course is built is C, assembly, and low-level data representation, but this is connected to higher levels (roughly how basic Java could be implemented), lower levels (the general structure of a processor and the memory hierarchy), and the role of the operating system (but not how the operating system is implemented).

This course develop students' sense of "what really happens" when software runs — and convey that this question can be answered at several levels of abstraction, including the hardware architecture level, the assembly level, the C programming level and the Java programming level. The core around which the course is built is C, assembly, and low-level data representation, but this is connected to higher levels (roughly how basic Java could be implemented), lower levels (the general structure of a processor), and the role of the operating system (but not how the operating system is implemented). For (computer science) students wanting to specialize at higher levels of abstraction, this could in the extreme be the only course they take that considers the "C level" and below. However, most will take a subset of Systems Programming, Hardware Design and Implementation, Operating Systems, Compilers, etc. For students interested in hardware, embedded systems, computer engineering, computer architecture, etc., this course is the introductory course after which other courses will delve both deeper (into specific topics) and lower (into hardware implementation, circuit design, etc.). The course has three principal themes [7,8]: (a) Representation: how different data types (from simple integers to arrays of data structures) are represented in memory, how instructions are encoded, and how memory addresses (pointers) are generated and used to create complex structures.(b) Translation: how high-level languages are translated into the basic instructions embodied in process hardware with a particular focus on C and Java. (c) Control flow: how computers organize the order of their computations, keep track of where they are in large programs, and provide the illusion of multiple processes executing in parallel.

2.2 Relationship of ABET criterion to course outcomes

This course supports the following seven course outcomes out of the outcomes required by ABET Criterion 3 for accrediting computer engineering programs shown in table 1 [10,11].

Table 1. Course Outcomes

Course Learning Outcomes	Outcome Indicators and Details	Assessment Methods and Metrics
O1. Ability to apply knowledge of mathematics, probability and engineering in microprocessor based system design	➢ Analysis of bus Fan-in and Fan-out requirements, analysis of bus and processor timing, performance evaluation, ➢ CPU execution time ➢ memory access time and bandwidth ➢ wait state computation ➢ computation of timing delays ➢ I/O performance such as interrupt latency and DMA speed	• Quizzes • Assignments • Exams
O2. Ability to design and conduct experiments related to microprocessor	➢ Design & conduct experiments on clock generation, power-on-reset generation, ready synchronization, address decoding, memory	• Lab work

based system design and to analyze their outcomes.	interfacing and I/O interfacing	
O3. Ability to design, debug and test a small scale microprocessor based system	➢ Design of Clock generation, Reset generation & synchronization, Wait state computation & generation, Ready synchronization, ➢ Address bus latching, data bus buffering, Design of Memory Map, Memory Address decoder, Memory Read and write logic ➢ Interfacing of RAM and EPROM memories. to processor(appropriate selection and connection of address bus, data bus, read/write control and chip select) Modes of I/O data transfer – Programmed or Polled I/O, Interrupt driven I/O, DMA ➢ Design of I/O Map, I/O address decoder and I/O Read and Write logic Interfacing of Parallel & Serial I/O devices to processor using peripheral chips 8255 PPI, 8254 PIT. - appropriate selection and connection of address bus, data bus, read/write control, chip select between processor and peripheral chips - data, control and status signal interconnections between peripheral chips and I/O devices - Programming of Peripheral interfacing chips ➢ debug and test the design as well as to develop small test program to test the design correctness and timing versus some requirements. ➢ Revise the design appropriately ➢ Report and document the design.	• Quizzes • Assignments • Exams • Lab work
O4. Ability to function as an effective team member	➢ Working in a team to design, assemble and test a small microprocessor based system prototype	• Lab work
O5. Ability to identify, formulate, and solve engineering problems in microprocessor based system design	➢ Identify, formulate and solve engineering problems in the microprocessor based system design considering the following : - Enhancements in the processor internal architecture, processor address & data bus width - Latest trends and developments in Memory Technology (SRAM, DRAM, SDRAM, RDRAM, DDR/DDR2) - Recent developments in I/O interfacing standards and I/O devices	• Quizzes • Assignments • Exams • Lab work
O6. Ability to use design tools for microprocessor system design, test and evaluation.	➢ Use of tools for debugging, develop techniques for testing, and use of trace analysis and timing for evaluation ➢ Use of Logic analyzers, oscilloscopes, logic probes, multimeters	• Lab work
O7. Ability to engage in self-learning	➢ Demonstrates reading and writing skills ➢ Identifying, retrieving, and organizing information ➢ Following a learning plan and demonstrate critical thinking skills such as applying the facts,	• Assignments • Quizzes • Exams

	formulae, theories, etc. to everyday situations	

2.3 Course content definition

After an extensive analysis of what prerequisite knowledge and skills for students to have before they graduated, the researchers identified and prepared a detailed analysis of the required knowledge and skills which are to be learned in each particular course. Included in the analysis is the recognition that everything presented cannot be expected to be mastered, that it is appropriate to simply introduce concepts and skills which will be studied more in depth in later courses.

The course content analysis resulted in a three-tier "learning frame." The researchers divided the knowledge and skills appropriate for the course into 1) mastery, 2) familiarity, and 3) exposure.

Mastery implies a thorough understanding of the concept or skill and the ability to identify and select this alternative as the best choice in a particular situation without prompting from the instructor. Familiarity requires a good understanding of the concept or skill and the ability to apply it to a similar situation or when guided to do so by the instructor. Exposure is an introduction to the concept or skill, at least to the extent of recognizing its existence and utility but not necessarily its application.

The curriculum committee is overseeing the process, but the entire faculty is engaged in the discussion and decision-making with our students offering significant insight. Many faculty members are devoting significant amounts of time to ensure that well-thought out, engaging and rigorous courses are available to the students.

2.4 Implementation

The revised course content is presently implemented in the first semester of school year 20012-2013. There are four sections handled by three faculty members and the researcher takes the lead role in the implementation. The strategy of team teaching, where the three faculty members work cooperatively with the same group of students, will be adopted in selected topics.

2.5 Present and future outlook

In recognition of the potential of developing researchers in the field hardware interfacing system, the researchers is currently undertaking a research work in the design and development of teaching modules of hardware interfacing system, in the process, an effort to identify additional hardware interfacing system instruments and tools to be acquired is underway. At present, the hardware interfacing system laboratory and methods study and work measurement laboratory are sharing on the room.

However, a separate hardware interfacing system laboratory will be established in the future. In addition, documentation/filming of work situations in selected local industries that need hardware interfacing system, improvements is being done. The documentation will lead to the study of the application of hardware interfacing system, in the local industries. Translating the findings into teaching modules enhances and reinforces the learning process.

2.6 Student comments

In developing the core courses, the researchers have asked and received many student comments. Students provided both formative and summative evaluation information for hardware interfacing system course concerning every type of activity -- every laboratory exercise, lecture slides and content, etc. The graduating class of May 2013 was the first to complete the entire new core curriculum. One year after graduation, we asked them to reflect on their undergraduate experience. The following are a sample of the comments the researchers received from those students who went into industry after graduation:

Q1. How would you compare your preparation for your job with others with Computer Engineering degree?
A1. Our program was definitely one of the best. In terms of classes offered and the amount of material we covered, there was no school that provided as much experience as the UVA program using C++ and Object Oriented Design.

A2. My first week on the job, I took a class with a recent XXX [major research university Computer Engineering graduate who had graduated at the top of her class. During the class, I was much more involved in what was going on and how things worked than this other employee and was able to make an immediate great impression on my new boss. ...

Q2. What parts of your UVA experience have been the most useful during your first year at work?

A1. Team work experience, having a good knowledge base on many hardware interfacing system topics (most other Computer engineering had good OS or PL experience whereas our curriculum covered most everything), and a good grasp on the entire software development cycle using OO design and C++ application.
A2. I think that the group projects have been the most valuable to me in my work experience thus far. On my project, everything is a collaborative effort between designers, developers, and testers. All of the classes (especially 340 [3SW]) really helped to prepare me for this type of group environment and how everyone must work together to achieve the goal.
A3. ... The most essential stuff has been having a thorough background in OO programming, data structures, and algorithms. But I'm in a much more programming intensive and research oriented position than a typical CS job. In fact, I'm surprised to say I actually found myself breaking out my Old probability and discrete math books (i.e. they were actually useful -- don't tell anyone I said this!) when I had to read a stack of graduate research papers in preparation for a project. ...

3 SUMMARY

The effort in enriching the learning experience of students in hardware interfacing system is an answer to the demands of present time for computer engineering graduates who are innovative, creative and concerned. Their role as computer engineer requires knowledge and concepts in the multidisciplinary field of hardware interfacing system to enable them to design work systems and workplaces that consider the well-being and efficiency of the organization.

Teaching hardware interfacing system in the traditional manner does not prepare the student as effectively as researchers would like. The researchers believe that a change in pedagogy provides the highest leverage for improving computer engineering education. With the revisions to this hardware interfacing system course, students are learning modern computing methods using practical tools and applications from the beginning. Continuing this philosophy throughout enhancing the course, the University of Auckland will produce computer professionals who are knowledgeable not only in content, but who have practical experience in the workings of the real-world environment.

REFERENCES

[1] Ganiron Jr, T.U., Martinez D.T. and Lacsamana C (2013), Development of Hardware Interfacing System for Visual C++, *International Journal of Advances in Applied Sciences*, Vol. 2, No. 4, pp. 1001-1005.

[2] Bazeghi, C (2006), General Information and Syllabus, Personal Computer Concepts: Software and Hardware, University of California, StaCruz.

[3] Chang, C. and Jifeng C (2001), Hardware/Software Interface Design, The United Nations University, Macau

[4] Kama, S. (2012, October), Evolution of the Trigger and Data Acquisition System in the ATLAS Experiment, In Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2012 IEEE, pp. 1787-1790.

[5] Vahid, F. and Givargis, T. (2002), Embedded System Design: A Unified Hardware/Software Introduction, Vol. 4, New York, NY: John Wiley & Sons.

[6] Vuletic, M., Pozzi, L., and Lenne, P. (2004, September), Programming Transparency and Portable Hardware Interfacing: Towards General-Purpose Reconfigurable Computing, In Application-Specific Systems, Architectures and Processors, 2004. Proceedings of 15th IEEE

International Conference, pp. 339-351.

[7] MacauVan R., Bolsens K, De Man I and Verkest, D. (1996, September),CoWare—A Design Environment for Heterogenous Hardware/Software Systems, In Proceedings of the Conference on European Design Automation, pp. 252-257, IEEE Computer Society Press.

[8] Brebner, G. (1996), A Virtual Hardware Operating System for the Xilinx XC6200, In Field-Programmable Logic Smart Applications, New Paradigms and Compilers, pp. 327-336, Springer Berlin Heidelberg.

[9] Ganiron Jr, T.U (2013, September 16-20), Accelerated Learning Techniques: Teaching Critical Thinking in Qassim University. *Journal of Proceedings of the 41st Annual Conference of the European Society for Engineering Education (SEFI)*, Leuven, Belgium.

[10] Computer Systems Engineering Courses (2014), The University of Auckland, New Zealand. Retrieved from http://web.ece.auckland.ac.nz/uoa/home/about/our-programmes-and-courses/our-courses/compsys-courses

[11] Ganiron Jr, T.U. (2013, September), Application of Accelerated Learning in Teaching Environmental Control System in Qassim University, *International Journal of Education and Learning*, Vol.2, No. 2, pp. 27-38.

[12] ABET Accreditation (2014), Baltimore, USA. Retrieved from http://www.abet.org/about-abet/