AC 2010-707: CLASSIFICATION AND ASSESSMENT OF PROJECTS IN COMPUTER ENGINEERING

Dick Blandford, University of Evansville

Dick Blandford is the department chair of the Department of Electrical Engineering and Computer Science at the University of Evansville. He received a PhD in EE from the University of Illinois.

Christina Howe, University of Evansville

Christina Howe is an assistant professor of Electrical Engineering at the University of Evansville. She received a PhD in EE from Vanderbilt University.

Anthony Richardson, University of Evansville

Tony Richardson is an associate professor of Electrical Engineering at the University of Evansville. He has a PhD in EE from Duke University

David Mitchell, University of Evansville

David Mitchell is an associate professor of Electrical Engineering at the University of Evansville. He holds a masters degree in Physics from the University of Toledo.

Classification and Assessment of Projects in Computer Engineering

Abstract

Computer engineering projects typically involve some combination of hardware and software. Students complete such projects in classes of the following subject matters: digital logic design, microcontrollers, digital systems, real-time systems, and less often digital controls or networks. More elaborate projects are done as a capstone experience. This paper is limited to non-capstone projects that include both hardware and software and are most frequently done in the sophomore and junior year of a computer engineering program.

Since such projects involve multiple areas within different disciplines, instructor's expectations, and work done over more than one year, it becomes difficult to assess how much of each topic a given student has covered. This paper suggests a way to classify such projects using the topics outlined in the Computer Engineering Body of Knowledge (BOK) document produced in 2004 by a joint task force of IEEE Computer Society and the ACM. A sample project is given along with its classification. Twenty-seven additional project summaries are provided. This data is used to determine how well the projects cover the required topics for purposes of assessment.

Introduction

At most universities, computer and electrical engineering students do labs in two different styles. In the "classic style", the student is given a lab sheet which describes in detail the "experiment" to be performed. It typically has a sequence of steps to be followed much like a recipe for baking a cake where the ingredients are the components and lab equipment. There is a second lab style which we call a "project lab" in which the student receives a paragraph or two describing some device to be designed. Such project labs typically include a bullet list of specifications and a grading scale with points assigned based on what specifications are met.

Both of these lab styles have clear advantages and their own peculiar drawbacks. The classic lab is very good for teaching students to use equipment such as an oscilloscope or a spectrum analyzer and for giving students practice in lab work that they are expected to understand as professionals. For the classic lab, it's easy to verify that any student who completes the lab has had experience with specified topics. For example, if a course objective is to provide students with experience in the use of a particular software tool such as MATLAB® or PSpice, a classic style lab where this software tool becomes one of the items on the "recipe list" will provide an easy-to-assess method of meeting this objective.

The project style lab is much more open-ended. Particular hardware and software tools are not specified, since the lab is described in terms of specifications to be met instead of methods which must be used to complete the lab. A student may choose to use a calculator or a custom computer program in place of MATLAB® for a project. Project labs do, however, provide the student with an experience that is closer to what that student will encounter in the professional workplace. In addition, students tend to remember better those experiences where they

personally devised a solution to a problem as opposed to those where they followed a script for a solution.

Assessment becomes problematic for project labs. Course objectives must be written in terms or project specifications instead of in terms of methods and tools that are used to solve a problem. Toward this end, we are proposing a method of classifying computer engineering projects to facilitate assessment and to clarify what course and curricular objectives are being met for particular projects.

The Computer Engineering Body of Knowledge (BOK)

Classification of projects needs to be done with some standards in mind. The ABET requirements for accreditation in computer engineering contain standards but these tend to apply to a curriculum and have insufficient detail to be used at the course level. The Computer Engineering Body of Knowledge[1,2,3] (BOK) document produced in 2004 by a joint task force of IEEE Computer Society and the ACM contains considerably more detail at the course level and we have relied on this document as a source for our classification standards.

The BOK attempts to list exhaustively all of the course requirements for an ABET accredited bachelor's degree in computer engineering. The BOK primary task force consisted of representatives from 22 Universities and had significant reviews from 27 other university faculty who were not on the task force. In putting the BOK together, the task force envisioned several versions of the computer engineering degree:

Implementation A – Computer Engineering Program Administered by a Computer Science Department.
Implementation B – Computer Engineering Program Administered by an Electrical and Computer Engineering Department.
Implementation C – Computer Engineering Program Administered Jointly by a Computer Science Department and a Department or College of Engineering.
Implementation D – Computer Engineering Program Representative of a Program in the United Kingdom and Other Nations.

The BOK task force developed a sample implementation for each version of the degree.

To make the BOK manageable, the task force defined 16 distinct areas in computer engineering and 2 additional areas in mathematics. The 18 areas are listed in Figure 1. Each area was broken down further into specific topics, and classroom lecture hours were assigned to each topic. For example, one area in computer engineering was titled "Programming Fundamentals" (Item 15 in Figure 1). This area was further broken down into topics as shown in Figure 2. All other knowledge areas were broken down in a similar fashion. See `http://www.eng.auburn.edu/ece/CCCE/` for a complete listing of topics in each area.

Project Classification

At the University of Evansville the junior and senior level labs are all project labs. We are interested in using the BOK data to classify these projects and to use that classification to better

understand what areas the labs are covering and which areas of the curriculum need more attention.

Number	Knowledge area
1	Discrete Structures [33 core hours]
2	Probability and Statistics [33 core hours]
3	Algorithms [30 core hours]
4	Computer Architecture and Organization [63 core hours]
5	Computer Systems Engineering [18 core hours]
6	Circuits and Signals [43 core hours]
7	Database Systems [5 core hours]
8	Digital Logic [57 core hours]
9	Digital Signal Processing [17 core hours]
10	Electronics [40 core hours]
11	Embedded Systems [20 core hours]
12	Human-Computer Interaction [8 core hours]
13	Computer Networks [21 core hours]
14	Operating Systems [20 core hours]
15	Programming Fundamentals [39 core hours]
16	Social and Professional Issues [16 core hours]
17	Software Engineering [13 core hours]
18	VLSI Design and Fabrication [10 core hours]

Figure 1
The 18 knowledge areas in the computer engineering BOK.

Programming Fundamentals [39 core hours]
History and overview [1]
Programming Paradigms [5]
Programming constructs [7]
Algorithms and problem-solving [8]
Data structures [13]
Recursion [5]
Object-oriented programming
Event-driven and concurrent programming
Using APIs

Figure 2
An example of an area in computer engineering and the topics included in that area

An instructor who wants to use an open-ended project for a class completes an online form which includes the course information, project title, a project description, and a checklist which classifies the project using the BOK classification data. This form becomes part of the data that is used for assessment of the course.

All courses are assessed on a three-year cycle. The assessment is done on an assigned assessment day at the end of the term and all faculty participate. A course which is being assessed will have a syllabus, text, a sample of student work from 4 to 6 students, and

classification forms for all open ended projects in the course. The assessment is done by two faculty who are familiar with the course material but who have not recently taught the course. These two faculty are responsible for checking the student work against the classification form to verify that the form is correct. This assessment process for courses is used in following terms to improve the course work.

As an example of how a project might be classified, consider a junior level course in microcontrollers, for which the prerequisite is a course in logic design and a course in the fundamentals of programming. A typical assignment requires that students design and implement a small project that uses a microcontroller in real time. One such project which has been used in the past is the *Digital Scale* project. The project description requires the design of a transducer, a microcontroller to read the transducer and translate the electrical signal into a human readable form, and a display device. The transducer uses an oscillator whose frequency is dependent on an inductance. The object to be weighed moves against a spring to push a metal slug into the inductor changing the frequency of the oscillator. A microcontroller is used to determine the frequency change and translate that change into a number representing the weight. The number is transmitted via a serial line to a computer which displays the result in a Graphical User Interface. Software on the microcontroller provided a way to zero the scale and do a calibration. The instructor filled out the classification form to as shown in Figure 3.

Area	Topic	Classification
Computer Systems Engineering	Requirements analysis and elicitation	1
	Specification	2
	Testing	2
	Project management	1
	Concurrent (hardware/software) design	3
	Implementation	3
	Reliability and fault tolerance	1
Circuits and Signals	Electrical Quantities	2
	Reactive Circuits and Networks	2
	Frequency Response	1
Digital Logic	Combinational logic circuits	2
	Modular design of combinational circuits	2
	Sequential logic circuits	2
	Digital systems design	3
	Design for testability	1
Electronics	Operational amplifiers	1
	Amplifier design	1
Embedded Systems	Embedded microcontrollers	3
	Embedded programs	3
	Reliable system design	1
	Tool support	2
	Interfacing and mixed-signal systems	2
Human-Computer Interaction	Graphical user interface	2
	I/O technologies	2
	Interactive graphical user-interface design	2
	Graphical user-interface programming	2

Figure 3
Classification of the *Digital Scale* project

The classification scale runs "zero" to "three" with "zero" meaning that the project does not contain that topic and "three" meaning the project uses that topic as a major focus.

Classification scale
0 – does not contain this topic
1 – this topic is introduced
2 – this topic is a significant part of this project
3 – this topic is a focus and a major part of this project

A sample classification form is shown in Figure 4. Figure 5 is a list and a brief description of the projects which we have classified using this system.

Results

Over the past year we have been able to classify about 30 projects and summarize their results over the curriculum to determine which items are not being covered and which areas should be shifted. Areas which were not covered well include Probability and Statistics, Algorithms, Database Systems, Social and Professional Issues, Software Engineering, and VLSI Design and Fabrication. Areas that had light coverage include Human-Computer Interaction, and Digital Signal Processing. The remaining areas were judged to be adequately covered, with a heavy emphasis on Embedded Systems and Digital Logic.

In our present curriculum, the Digital Signal Processing course is an elective for the computer engineering students but is required for the electrical engineering majors. Likewise, the Human-Computer Interface course is an elective for both computer engineering and computer science majors. Computer engineering faculty are presently considering whether the curriculum needs to be altered to place more emphasis on these two courses. It seems likely that we will require the course on Human-Computer interfacing and make the presently required course on Programming Languages an elective. A revision of the linear systems sequence is being considered to provide more emphasis on Digital Signal Processing.

It seems unlikely that we will ever have open ended projects related to Probability and Statistics, Algorithms, Database Systems, Social and Professional Issues, or Software Engineering. These topics will be left for coverage in the senior capstone project or in other coursework. Our program provides an introduction to VLSI Design and Fabrication and makes use of simulation but no open-ended projects in that area.

We are continuing to update our classification system and to validate the classifications that have been done by means of assessments. We believe this system is a valuable tool that allows us to look at our curriculum from a new viewpoint.

Algorithms		**Digital Signal Processing (Cont)**		**Operating Systems (Cont)**	
Basic algorithmic analysis		Sampling		Security and protection	
Algorithmic strategies		Transforms		File systems	
Computing algorithms		Digital filters		System performance evaluation	
Distributed algorithms		Discrete time signals		**Computer Systems Engineering**	
Algorithmic complexity		Window functions		Life cycle	
Basic computability theory		Convolution		Requirements analysis and elicitation	
Computer Architecture and Org		Audio processing		Specification	
Fundamentals of computer architecture		Image processing		Architectural design	
Computer arithmetic		**Electronics**		Testing	
Memory system organ. and architecture		Electronic properties of materials		Maintenance	
Interfacing and communication		Diodes and diode circuits		Project management	
Device subsystems		MOS transistors and biasing		Concurrent design	
Processor systems design		MOS logic families		Implementation	
Organization of the CPU		Bipolar transistors and logic families		Specialized systems	
Performance		Design parameters and issues		Reliability and fault tolerance	
Distributed system models		Storage elements		**Social and Professional Issues**	
Performance enhancements		Interfacing logic families and buses		Public policy	
Computer Systems Engineering		Operational amplifiers		Methods and tools of analysis	
Life cycle		Circuit modeling and simulation		Prof. and ethical responsibilities	
Requirements analysis and elicitation		Data conversion circuits		Risks and liabilities	
Specification		Electronic voltage and current sources		Intellectual property	
Architectural design		Amplifier design		Privacy and civil liberties	
Testing		Integrated circuit building blocks		Computer crime	
Maintenance		**Embedded Systems**		Economic issues in computing	
Project management		Embedded microcontrollers		Philosophical frameworks	
Concurrent (hardware/software) design		Embedded programs		**Software Engineering**	
Implementation		Real-time operating systems		Software processes	
Specialized systems		Low-power computing		Software requirements and spec	
Reliability and fault tolerance		Reliable system design		Software design	
Circuits and Signals		Design methodologies		Software testing and validation	
Electrical Quantities		Tool support		Software evolution	
Resistive Circuits and Networks		Embedded multiprocessors		Software tools and environments	
Reactive Circuits and Networks		Networked embedded systems		Language translation	
Frequency Response		Interfacing and mixed-signal systems		Software project management	
Sinusoidal Analysis		**Human-Computer Interaction**		Software fault tolerance	
Convolution		Foundations of human-computer interact		**VLSI Design and Fabrication**	
Fourier Analysis		Graphical user interface		Electronic properties of materials	
Filters		I/O technologies		Function of the inverter structure	
Laplace Transforms		Intelligent systems		Combinational logic structures	
Database Systems		Human-centered software evaluation		Sequential logic structures	
Database systems		Human-centered software development		Semiconductor memories & array structures	
Data modeling		Interactive graphical user-interface design		Chip input/output circuits	
Relational databases		Graphical user-interface programming		Processing and layout	
Database query languages		Graphics and visualization		Circuit characterization and performance	
Relational database design		Multimedia systems		Alternative ckt structures/low power design	
Transaction processing		**Computer Networks**		Semi-custom design technologies	
Distributed databases		Communications network architecture		ASIC design methodology	
Physical database design		Communications network protocols		**Discrete Structures**	
Digital Logic		Local and wide area networks		Functions, relations, and sets	
Switching theory		Client-server computing		Basic logic	
Combinational logic circuits		Data security and integrity		Proof techniques	
Modular design of combinational circuits		Wireless and mobile computing		Basics of counting	
Memory elements		Performance evaluation		Graphs and trees	
Sequential logic circuits		Data communications		Recursion	
Digital systems design		Network management		**Probability and Statistics**	
Modeling and simulation		Compression and decompression		Discrete probability	
Formal verification		**Operating Systems**		E-PRS2 Continuous probability	
Fault models and testing		History and overview		Expectation	
Design for testability		Design principles		Stochastic Processes	
Digital Signal Processing		Concurrency		Sampling distributions	
Theories and concepts		Scheduling and dispatch		Estimation	
Digital spectra analysis		Memory management		Hypothesis tests	
Discrete Fourier transform		Device management		Correlation and regression	

0 – does not contain this topic, 1 – topic is introduced, 2 – topic is a significant part of project, 3 – topic is a focus and a major part of this project

Figure 4

Sample classification form.

Project	Description
Digital Alarm Clock	Displays hour and minutes, battery operated and self contained. Used an 8-bit microcontroller and an LCD display. Display can be in base 10, binary, or hex.
Sliding Scale Clock	Tells time with two sliding bars driven by stepper motors. Can be done with Lego parts.
Paint Ball Gun Muzzle Velocity	Measures and displays the muzzle velocity of a paint ball gun. Uses an 8-bit microcontroller and an LCD display.
Multiplexed Rotating Line Display	A vertical line of LEDs rotates on the end of an arm driven by a fan motor. A microcontroller flashes the LEDs in synchronism to display a message.
Capacitance meter	Measures capacitance by charging the capacitor with a current source and measuring the voltage in time. Displays results on an LCD display and has self adjusting scale.
Coin counter	Counts coins by allowing them to fall past a light sensor on a ramp. The time the light is blocked is proportional to the coin size. Sorts coins into various bins depending on denomination and displays result on an LCD display.
Function Generator	Microcontroller is used to generate triangle, ramp, square, and sinusoidal waves at user adjustable frequencies.
Acceleration of a Moving Object	Measures and displays the acceleration of a moving object. Uses an 8-bit microcontroller and an LCD display.
Infrared Bi-directional Communication	Uses two microcontrollers to receive and transmit audio over an infrared beam over a 10 foot span.
Microcontroller Low-pass Digital Filter	Uses a 32-bit ARM processor to make a user-adjustable low pass filter.
Light Tracking Servo-motor	Uses an 8-bit microcontroller to sense a light source and track it with a servo motor. The light source is modulated.
Oven Temperature Controller	Uses an 8-bit microcontroller to control the temperature of a Styrofoam box heated by a light bulb.
Altimeter	A balloon expands or contracts with changing air pressure which is related to altitude. This size change moves a slug in or out of an inductor and the microcontroller determines the frequency of the inductor based oscillator and translates this data to altitude. Uses an 8-bit microcontroller and an LCD display. Battery operated and self contained.
Digital Scale	A weight is applied to a spring which allows a slug to move with respect to an inductor which changes the frequency of the inductor based oscillator. An 8-bit microcontroller translates this data to weight and sends it via a serial line to a PC running C# with a GUI user interface.
Etching and Sketching	Implements a program using a real time operating system to imitate Etch A Sketch. Leaves a trail of * on the screen. Can have multiple screens and imitates all of the actual physical functions.
TCP Time/Date server	Uses a NetBurner microcontroller running RTEMS to provide the time and date to a client on the net.
Embedded Web Programming	Uses a NetBurner microcontroller to provide a web page displaying database information to a client on the net.
Pulse Train Generator/ Monitor	Desktop computer interfaces with a microcontroller. User inputs frequency and duty cycle which is transmitted to the microcontroller to produce. In monitor mode, microcontroller sends frequency and duty cycle to desktop for display.
Magnetic Levitation	Levitates a small magnet beneath an electromagnet. Has an optical sensor and uses a microcontroller to drive the electromagnet with PWM.

Figure 5

Summary of open-ended projects which have been classified using the BOK data.

Project	Description
Marble Sorter	Sorts brass, steel, and nylon marbles into appropriate bins. Uses an 8-bit microcontroller and displays status on an LCD screen.
Phased Array in Sound	Five small equally spaced speakers are drive by a microcontroller with the same signal but different phases. Illustrates phased array beam steering. Very dramatic effect. This can also be done with microphones.
Very Remote Sensor Pack	A sensor pack with a microcontroller is programmed via a PC and placed at a remote location for up to one year where it wakes up on occasion and collects data. Data is recovered via a PC.
Line Scan Racetrack Camera	Uses a thin line of light sensors to take a picture of a finish line as it is crossed. Microcontroller monitors image and stores only the image that catches the finish.
Spectrum Analyzer	Uses a 32-bit ARM processor to capture an audio clip, do the FFT, and display the result on a PC.
Universal Stepper-motor Controller	This is a bipolar stepper motor driver that is microcontroller based and allows the user to control the step rate, acceleration, direction, current, and step mode (half stepping and micro stepping) via a PC interface.
USB Sensor	A microcontroller with a USB interface transmits requested sensor data to a PC via the USB port. Student must write USB driver for microcontroller and DLL to run under C# on the PC in addition to the GUI user interface.
CAN Sensor	A remote microcontroller sends sensor data via the CAN bus to a second microcontroller which provides an interface to a PC where the data is displayed.

Figure 5 (Cont)

Summary of open ended projects which have been classified using the BOK data.

Bibliography

1. Soldan, D.L.; Nelson, V.P.; McGettrick, A.; Impagliazzo, J.; Srimani, P.; Theys, M.D.; Hughes, J.L.A. Frontiers in Education, 2004. FIE 2004. 34th Annual Volume, Issue 20-23 Oct. 2004.

2. A pdf version of the BOK can be downloaded from http://www.eng.auburn.edu/ece/CCCE/

3. Nelson, Victor P., Soldan, David L., McGettrick, Andrew, Impagliazzo, John, Srimani, Pradip, Theys, Mitchell D. and Hughes, Joseph L. A., COMPUTING CURRICULUM - COMPUTER ENGINEERING (CCCE) A MODEL FOR COMPUTER ENGINEERING CURRICULA IN THE NEXT DECADE, ASEE 2004 Annual Conference, Pittsburgh.