Response to CAC's Draft Statement

Department of Computer Science, UC Riverside

February 21, 2007

Abstract

This document details the changes made to the CS curriculum to address the weakness in coverage of social and ethical issues in computing (Standard IV-17) mentioned in CAC's Draft Statement regarding its 2006 review of the Computer Science degree program at UC Riverside.

1 Relevant Portions of Draft Statement

This document is in response to the following paragraph on page 4 of CAC's Draft Statement regarding the 2006 review of the Computer Science program at the University of California, Riverside.

> The program meets the intent of the Curriculum Category by satisfying all associated standards. However, there is a concern with respect to Standard IV-17 that constitutes a weakness with respect to the Curriculum Category.

In its conclusion on page 9, the Draft Statement elaborates on this point as follows:

> However, there is a weakness with respect to the Curriculum Category. This weakness is most closely associated with Standard IV-17. Specifically,
>
> 1. (Standard IV-17) Adequate coverage of social and ethical issues in computing must be assured and the current lack of faculty consensus in addressing these topics results in significant uncertainty that adequate coverage of this topic will be sustained.

The weakness may affect the stability, overall quality, or future accreditation of the program and will be of special interest to the next visiting team.

2 Action Taken

To address this weakness, UC Riverside's Department of Computer Science and Engineering has:

- expanded one of the CS program's outcomes, namely outcome J, which previously read: "*a knowledge of contemporary issues*," to now read: "*a knowledge of contemporary issues,* **including ethical and social issues**." [emphasis added],
- made outcome J an explicit objective of four upper-division (*i.e.* junior/senior-level) required courses:[1]
  - CS 152 (Compiler Design)
  - CS 153 (Design of Operating Systems)
  - CS 161 (Design and Architecture of Computer Systems)
  - CS 179 (Project in Computer Science),
- modified the syllabi of those four courses to explicitly note the aspects of "ethical and social issues" to be covered in each case,
- and prepared an instructor's manual with a taxonomy of social and ethical topics to cover.

2.1 Rationale

The CS&E faculty believe that students are best served by teaching ethical and social issues alongside their regular course work. If, instead, a separate course were to be added to the curriculum to address these instructional points, the students would view ethical and social issues as separate from

[1]Each course has say five to ten *course objectives*, which are related to program outcomes via a *course matrix*. Course objectives are areas of knowledge and/or skill that students should have mastered by the end of the course. These should not be confused with *Program Educational Objectives*, which per EAC are "*broad statements that describe the career and professional accomplishments that the program is preparing graduates to achieve.*"

the rest of computer science. By teaching them alongside technical material, instructors can link the technical and social concepts, provide more germane examples, and encourage students to view both ethical and social issues, as well as technical issues, as critical to the engineering-design process.

Placing explicit reference to ethical and social issues in the program outcomes and the course-level objectives of four upper-division courses stresses to all constituents that the program addresses these issues. To mandate that instructors cover these issues, via classroom time and testing resources, within these four courses, the syllabus of each course now explicitly reflects this objective.

Finally, to ensure some consistency of coverage across offerings of a given course and across the program, an instructor manual with specifics on instruction and coordination among courses is provided to all instructors. It provides suggestions on topics and methods for instruction and testing comprehension.

Instructors are charged with testing for each course objective on the final exam or with essays. Those per-question scores are individually recorded and assessed as part of the departmentwide assessment process that measures the degree to which program outcomes are being achieved.

3 Supporting Materials

The following supporting materials are attached.

- an updated list of program outcomes.
- updated syllabi and course objectives for CS 152, CS 153, CS 161, and CS 179.
- the current draft of the instructor's manual.[2]

[2]We expect this manual to be updated with additional examples and areas of social and ethical concern.

ATTACHMENT: UPDATED LIST OF COMPUTER SCIENCE PROGRAM OUTCOMES

The following are the program outcomes for UCR's B.S. degree program in Computer Science:

(a) an ability to apply knowledge of mathematics, science, and engineering

(b) an ability to design and conduct experiments, as well as to analyze and interpret data

(c) an ability to design a system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability

(d) an ability to function on multi-disciplinary teams

(e) an ability to identify, formulate, and solve engineering problems

(f) an understanding of professional and ethical responsibility

(g) an ability to communicate effectively

(h) the broad education necessary to understand the impact of engineering solutions in a global, economic, environmental, and societal context

(i) a recognition of the need for, and an ability to engage in life-long learning

(j) a knowledge of contemporary issues, including social and ethical issues

(k) an ability to use the techniques, skills, and modern engineering tools necessary for engineering practice.

COURSE DESCRIPTION

Dept., Number	Computer Science and Engineering, CS 152	Course Title	Compiler Design
Units	4	Course Coordinator	Tom Payne
Required/elective	required	URL (if any):	

Current Catalog Description: Covers the fundamentals of compiler design, including lexical analysis, parsing, semantic analysis, compile-time memory organization, run-time memory organization, code generation, and compiler portability issues. Laboratory work involves exercises covering various aspects of compilers.

Textbook: *Modern Compiler Implementation in Java* (second edition) by Andrew Appel and Jens Palsberg

References/Materials

The web site for the text, http://www.cs.princeton.edu/~appel/modern/java/. Also, when Tom Payne teaches the class, the first seven chapters (75 pages) of his notes on Compiler Design, which he distributes to the students in postscript, dvi, and html formats.

Course Goals/Objectives:

1. Provide students with a basic understanding of the design and functionality provided by compilers and interpreters, including theoretical foundations as far as necessary
2. Provide students with practical experience building a compiler for a (small) imperative programming language, ideally generating code for an actual machine
3. Using compilers, a well-explored field from the perspective of software engineering, illustrate various useful design and implementation techniques, focusing on object-oriented ones
4. Provide students with an understanding of contemporary social and ethical issues in computer science

Prerequisites by Courses and Topics: CS 061: Machine Organization and Assembly Language Programming; CS 141: Intermediate Data Structures and Algorithms; CS 150: Theory of Automata and Formal Languages

Major Topics Covered in the Course

- Automatic generators for lexical analyzers (scanners): review of the set-of-states construction for determinizing nondeterministic finite automata, use of ordered EBNF to describe lexical categories, lookahead, left-context, case-study LEX, compression of state table.
- Automatic generators for LALR parsers: Converting context-free grammars (BNF) to "railroad diagrams" to nondeterministic PDAs. Determinizing NPDAs: LR(0) tables, LR(1) tables, generating LALR tables from LR(1) table, LALR table by identifying states during construction of LR(1) table and propagation of lookaheads.
- Syntax-directed translation.
- The run-time environments including allocation and accessing of static, dynamic and automatic objects.
- Contemporary social and ethical issues (computer crime and software integrity)

Laboratory schedule: number of sessions per week and duration of each session:
Lecture, 3 hours; laboratory, 3 hours

Laboratory projects (specify number of weeks on each)

One week of orientation to the term project, which is to construct a compiler for the MiniJava language specified at the back of the textbook.
One week on the use of LEX (FLEX) and generating scanner for term project.
One week on the use of YACC (BISON) and generating parser for term project.
One week on the building of the syntax tree for the term project.
Two weeks on semantic analysis and checking for the term project.
Three weeks on code generation for the term project.

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	5.00%		Data Structures	5.00%	
Software Design	24.00%		Prog. Languages	33.00%	33.00%
Comp. Arch.					

Oral and Written Communications:
Every student is required to submit at least __0___ written reports (not including exams, tests, quizzes, or commented programs) of typically __0___ pages and to make ___0__ oral presentations of typically ___0__ minutes duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

Students have at least one hour of lecture on contemporary social and ethical issues, focusing on software integrity and computer crime. Students are tested with an essay, either in class or on the final exam.

The students are lectured on the importance of high-level programming languages to programmer productivity and the importance of programmer productivity to all of information technology.

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

This course is where the material such as context free grammars and syntax trees is put to practical use in building important tools. Specifically, the set-of-states construction learned in the theory of computation class is used in the construction of scanner generators and parser generators. (That construction is covered only in class. Then the students use such generators in the development of the scanner and parser for their term project.)

Problem Analysis

Please describe the analysis experiences common to all course sections.

The analysis and design for the overall framework for the term project is presented in class and follows the presentation in the text. The analysis and design for some of the components (semantic analyzers and code generators for the individual expressions, declarations, and statements) are presented in class, but the students must do the rest on their own, with some in-lab help from the TAs.

Solution Design

Please describe the design experiences common to all course sections.

Part of the description of design experience is given in the box above. Beyond that, we tend to emphasize object-oriented, test-driven design. We recommend that students treat each type of node in the syntax tree as a class, with the various kinds of expressions being subclasses of the Expression class, etc. Each of those classes must have a constructor, a method for doing semantic analysis, and a method for generating code. So, each class become a nicely constrained design problem.

Assessment methods: Final exam: 34%; Project: 33%; Quizzes: 33%.

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

The design and implementation of a compiler is a significant design experience and teaches a lot about good coding and testing practices. But in CS152 the training wheels are on. Students are presented with much of the top-level analysis and design, and they are coached a lot on how to stay out of trouble. This is realistic preparation for the capstone design courses, in which the training wheels are off.

Relationship of course to program outcomes: The contribution of CS152 to program outcomes (a)-(k) or (1) – (3) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
Provide students with a basic understanding of the design and functionality provided by compilers and interpreters, including theoretical foundations as far as necessary	1	0	0	0	0	0	0	0	0	0	3
Provide students with practical experience building a compiler for a (small) imperative programming language, ideally generating code for an actual machine	3	0	3	0	1	0	0	0	0	0	3
Using compilers, a well-explored field from the perspective of software engineering, illustrate various useful design and implementation techniques, focusing on object-oriented ones	3	0	2	0	1	0	0	0	0	0	3
Provide students with an understanding of contemporary social and ethical issues	0	0	0	0	0	0	0	0	0	3	0

Prepared by, and date of preparation:
Tom Payne 6/18/06

COURSE DESCRIPTION

Dept., Number	Computer Science and Engineering, CS 153	Course Title	Design of Operating Systems
Units	4	Course Coordinator	Vana Kalogeraki
Required/elective	Required	URL (if any):	http://www.cs.ucr.edu/~vana/cs153

Current Catalog Description

Principles and practice of operating system design, including concurrency, memory management, file systems, protection, security, command languages, scheduling, and system performance. Laboratory work involves exercises covering various aspects of operating systems

Textbook

Operating System Concepts by Avi Silberschatz, Peter Baer Galvin, Greg Gagne, John Wiley & Sons, Sixth Edition
Kernel Projects for Linux by Gary Nutt, Addison Wesley

References/Materials

http://www.cs.ucr.edu/~vana/cs153/resources.htm Additional resources for the course
http://www.cs.ucr.edu/~vana/cs153/assignments.htm Supporting material for the project assignments

Course Goals/Objectives

1. Study basic principles underlying the design of operating systems with a focus on principles and mechanisms used throughout the design
2. An understanding of CPU scheduling, storage management: memory management, virtual memory and file systems
3. Study of concurrency control and synchronization, classic algorithms for synchronization and concurrency management
4. Study Deadlocks Devices, device management and I/O systems
5. Study dynamic binding
6. An understanding of protection, access control and security
7. Improve skills in concurrent programming and introduce kernel programming
8. Provide students with an understanding of contemporary social and ethical in computer science

Prerequisites by Courses and Topics

CS 061. Machine Organization and Assembly Language Programming
CS 141. Intermediate Data Structures and Algorithms
C++ programming proficiency

Major Topics Covered in the Course

Introduction to Operating Systems, Computer-System Structures and Operating-System Structures. Process Management: Processes, Threads, CPU Scheduling, Process Synchronization and Deadlocks. Storage Management: Memory Management, Virtual Memory, File-System Interface, File-System Implementation, Contemporary Social and Ethical Issues (privacy and information use)

Laboratory schedule: number of sessions per week and duration of each session

Lecture 3 hours
Laboratory 3 hours

Lab1: Practicing system functions
Lab2: Working on the 1st project assignment - shell programming
Lab3: Working on the 1st project assignment - shell programming
Lab4: Introduction to multi-threading
Lab5: Working on the 2nd project assignment – multithreading
Lab6: Working on the 2nd project assignment – multithreading
Lab7: Introduction to file systems
Lab8: Working on the 3rd project assignment – file systems
Lab9: Working on the 3rd project assignment – file systems
Lab10: Working on the 3rd project assignment – file systems

Laboratory projects (specify number of weeks on each)

1st project – Shell programming: 2 weeks
2nd project – Multithreading programs: 2 weeks
3rd project – File system: 3 weeks

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	10	10	Data Structures	10	5
Software Design	20	15	Prog. Languages	15	5
Comp. Arch.	5	5			

Oral and Written Communications:

Every student is required to submit at least __1__ written reports (not including exams, tests, quizzes, or commented programs) of typically _2-3__ pages and to make __1__ oral presentations of typically __10_ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

Students have at least one hour of lecture on contemporary social and ethical issues, focusing on privacy and information use. Students are tested with an essay, either in class or on the final exam.

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

Introduction to Operating Systems Structures – 1 lecture
Introduction to Computer System Structures – 1 lecture
Process Management – 2 lectures
Threads – 2 lecture
CPU Scheduling - 2 lectures
Process Synchronization – 2 lectures
Deadlocks – 3 lectures
Memory Management – 3 lectures
Virtual Memory – 2 lectures
File systems – 2 lectures

Problem Analysis

Please describe the analysis experiences common to all course sections.

Students will learn to analyze the requirements and goals of the projects, identify the important components and analyze the efficiency of their solutions.
The students will learn the importance of analysis through hands-on experience.

Solution Design

Please describe the design experiences common to all course sections.

A primary goal of the course is the design of solutions that meet the project requirements. The students will design, implement and test various components of an operating system (a UNIX Shell, a CPU Scheduler, a Multithreading program and a File System). The students will apply software tools to build and evaluate their designs.
The students will learn the importance of solution design through hands-on experience. This will help them understand the complexity of building real systems.

Assessment methods

Homeworks – 10%
Projects – 30%
2 Midterm Exams – 20%
Final Exam – 40%

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

Students will learn how to design and implement various components of an operating system, thus they will understand the overall engineering design process (identification of constraints, description of design criteria and objectives, usage of tools, development of a prototype, evaluation of the prototype based on the design criteria). Students will get hands-on experience in proposing, designing and executing the project.
Students will also learn how to work in teams; each student will actively participate as a member of a team, collaborate with the other team members, share the daily design activities and management of the project and contribute to achieve the project goals. At the end of the project, they will have to write a team report that describes their design, thus they will learn how to communicate their ideas effectively.

Relationship of course to program outcomes: The contribution of CS153 to program outcomes (a)-(k) or (1) – (13) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
Study basic principles underlying the design of operating systems with a focus on principles and mechanisms used throughout the design	1	0	1	0	1	0	0	0	0	0	3
An understanding of CPU scheduling storage management: memory management virtual memory and file systems	1	0	0	0	0	0	0	0	0	0	3
Study of concurrency control and synchronization classical algorithms for synchronization and concurrency management	1	0	1	0	0	0	0	0	0	0	3
Study Deadlocks Devices device management and I/O systems	1	0	0	0	0	0	0	0	0	0	3
Study dynamic binding	0	0	0	0	0	0	0	0	0	0	3
An understanding of protection access control and security	1	0	0	0	0	1	0	1	0	0	3
Improve skills in concurrent programming and introduce kernel programming	3	2	0	0	0	0	0	0	0	0	3
Provide students with an understanding of contemporary social and ethical issues	0	0	0	0	0	0	0	0	0	3	0

Prepared by, and date of preparation:
Vana Kalogeraki – 06/15/06

COURSE DESCRIPTION

Dept., Number	Computer Science and Engineering, CS 161	Course Title	Design and Architecture of Computer Systems
Units	4	Course Coordinator	Dr. Laxmi N. Bhuyan
Required/elective	Elective	URL (if any):	http://www.cs.ucr.edu/~bhuyan/cs161 (lectures) http://www.cs.ucr.edu/~vladimir/cs161 (discussions)

Current Catalog Description: A study of the fundamentals of computer design. Topics include the performance evaluation of microprocessors, instruction set design and measurements of use, microprocessor implementation techniques including multi-cycle and pipelined implementations, computer arithmetic, memory hierarchy, and input/output (I/O) systems.

Textbook: Patterson and Hennessy, "Computer Organization and Design" Morgan Kaufmann publisher

References/Materials

Lecture slides, available via the lectures web site; selected discussion notes, available via the discussions web site.

Course Goals/Objectives:

1. Understand instructions as the language of the machine and the tradeoffs in instruction set design
2. Introduction to the issues and factors that impact performance, both hardware and software
3. Learn how to design the data-path and control unit as the heart of the CPU
4. Introduction to computer arithmetic: fast addition and multiplication
5. Introduction to memory hierarchy: simple caches and virtual memory
6. Learn how to design fast CPUs using pipelining
7. Introduction to advanced processors using instruction level parallelism
8. Provide students with an understanding of contemporary social and ethical in computer science

Prerequisites by Courses and Topics: CS 120B/EE 120B: Introduction to Embedded Systems. Introduction to hardware and software design of digital computing systems embedded in electronic devices (such as digital cameras or portable video games). Topics include custom and programmable processor design, standard peripherals, memories, interfacing, and hardware/software tradeoffs. Laboratory involves use of synthesis tools, programmable logic, and microcontrollers and development of working embedded systems; concurrent enrollment in CS 161L.

Major Topics Covered in the Course

Chapter 1: Introduction
Chapter 2: MIPS Instructions
Chapter 3: Computer Arithmetic
Chapter 4: Understanding Performance
Chapter 5: The Processor: Datapath and Control
Chapter 7: Memory Hierarchy
Chapter 8: I/O System
Contemporary social and ethical issues

Laboratory schedule: number of sessions per week and duration of each session:

There is a separate laboratory class, CS 161 L, which is a co-requisite for the course. In addition to the 3 hours a week of lectures, this course has 1 weekly discussion session, which lasts for 50 minutes.

Weekly Discussion Schedule:

Week 1: Introduction
Week 2: Homework 1 (**Assessing and Understanding Performance**) introduced
Week 3: Solutions for Homework 1, Homework 2 (**Instructions**) introduced
Week 4: Homework 2 discussed
Week 5: Solutions for Homework 2, Homework 3 (**Arithmetic**) introduced
Week 6: Homework 3 discussed
Week 7: Solutions for Homework 3, Homework 4 (**Datapath and Control**) introduced
Week 8: Homework 4 discussed
Week 9: Solutions for Homework 4, Homework 5 (**Memory Hierarchy**) introduced
Week 10: Overview, Solutions for Homework 5

Laboratory projects (specify number of weeks on each)

There were no specific projects for the discussion sessions.

Instead, there were 5 homeworks, and discussions were structured to present material and strategies useful for solving the homeworks. Solutions to selected problems were presented after homeworks have been graded.

On average, for each of the 5 homework topics two weeks were allocated:

- Assessing and Understanding Performance (2 weeks)
- Instructions (2 weeks)
- Arithmetic (2 weeks)
- Datapath and Control (2 weeks)
- Memory Hierarchy (2 weeks)

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms			Data Structures		
Software Design			Prog. Languages		
Comp. Arch.	100				

Oral and Written Communications:
Every student is required to submit at least __0___ written reports (not including exams, tests, quizzes, or commented programs) of typically _0____ pages and to make __0___ oral presentations of typically _0____ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

Students have at least one hour of lecture on contemporary social and ethical issues, focusing on property rights. Students are tested with an essay, either in class or on the final exam.

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

The theoretical material consists of understanding of the instruction sets of a typical CPU, basic building blocks of datapath, and memory units. Approximately 50% of the time is spent in explaining these operations.

Problem Analysis

Please describe the analysis experiences common to all course sections.

Analyzing the problem to get the understanding of the requirements was emphasized in the lectures and in the discussions.

Solution Design

Please describe the design experiences common to all course sections.

This course emphasizes quantitative approach to computer architecture; hence, a significant number of problems required quantitatively evaluating several alternative solutions to find the best one for the problem at hand.

Assessment methods:

20% Homeworks (5 homeworks, 4% each)

20% Exam 1

25% Exam 2

35% Exam 3

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

Relationship of course to program outcomes: The contribution of CS161 to program outcomes (a)-(k) or (1) – (7) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
Understand instructions as the language of the machine and the tradeoffs in instruction set design	1	1	1	0	1	0	0	0	0	0	3
Introduction to the issues and factors that impact performance, both hardware and software	2	2	2	0	1	0	0	0	0	0	3
Learn how to design the data-path and control unit as the heart of the CPU	3	2	3	0	0	0	0	0	0	0	3
Introduction to computer arithmetic: fast addition and multiplication	3	0	3	0	2	0	0	0	0	0	3
Introduction to memory hierarchy: simple caches and virtual memory	2	2	3	0	2	0	0	0	0	0	3
Learn how to design fast CPUs using pipelining	3	2	3	0	0	0	0	0	0	0	3
Introduction to advanced processors using instruction-level parallelism	3	2	3	0	2	0	0	0	0	0	3
Provide students with an understanding of contemporary social and ethical issues	0	0	0	0	0	0	0	0	0	3	0

Prepared by, and date of preparation:
L.N. Bhuyan, June 22, 2006

COURSE DESCRIPTION

Dept., Number	CS 179	Course Title	Project in Computer Science
Units	4	Course Coordinator	Christian Shelton
Required/elective	required	URL (if any):	

Current Catalog Description

Under the direction of a faculty member, students (individually or in small teams with shared responsibilities) propose, design, build, test, and document software and/or hardware devices or systems. Requires a written report, giving details of the project and test results, and an oral presentation of the design aspects. Emphasizes teamwork, making technical presentations, and developing oral and written communication skills.

Textbook

varies from offering to offering

References/Materials

ACM Code of Ethics and Professional Responsibility, other topical information, plus additional material that varies from offering to offering.

Course Goals/Objectives

1: Balancing design tradeoffs: cost performance schedule and risk
2: Writing project proposals
3: Team-project organization and management (including time lines)
4: Requirements capture and analysis
5: Design and architecture
6: Prototyping (possibly via simulation)
7: Verification/validation
8: Writing and presenting final reports
9: Engineering professionalism and responsibility
10: Engineering careers and the modern world
11: Comtemporary social and ethical issues in computer science

Prerequisites by Courses and Topics

CS 141 with a grade of "C-" or better; ENGR 180; 12 additional upper-division units in Computer Science. Each offering of CS179 is oriented toward a particular topic such as compilers, in which case it would have our upper-division course on compilers, CS152, as a prerequisite. Similarly for all other topics.

Major Topics Covered in the Course

teamwork, technical presentations, oral and written communication, various technical topics as per course offering

Laboratory schedule: number of sessions per week and duration of each session

nine lab-hours per week, three of which are rigidly scheduled per week, the remainder as desired by the students

Laboratory projects (specify number of weeks on each)

project proposal (2 weeks), design specification (2 weeks), final project (10 weeks)

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms		10%	Data Structures		
Software Design		15%	Prog. Languages		
Comp. Arch.					

Oral and Written Communications:

Every student is required to submit at least __1___ written reports (not including exams, tests, quizzes, or commented programs) of typically __20___ pages and to make __1___ oral presentations of typically __15___ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

Professional codes of ethics, social implications of created artifacts including current modern systems. Discussion of current systems in the topic area. Emphasis on responsibility and social implications. Essay analyzing social and ethical issue required of each student.

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

Varies by topic. Usually advanced algorithms or architectures required for design project.

Problem Analysis

Please describe the analysis experiences common to all course sections.

Students must take a user-specified task and break it into engineering components.

Solution Design

Please describe the design experiences common to all course sections.

Students must take the components and design, specify, and build each one to complete the final project.

Assessment methods

Preliminary Specifications (20%), Teamwork/weekly reports (20%), Final Project (20%), Final Report (20%), Final Presentation (20%)

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

This course directly involves the student in the identification of an engineering problem, the design and specification of a system to meet the problem's needs, the implementation of the solution, and the presentation of the results. Throughout the course, the students work in teams and must deal with team dynamics and scheduling. Students are responsible for making periodic "reports" to the project manager (the instructor).

Relationship of course to program outcomes: The contribution of CS179 to program outcomes (a)-(k) or (1) – (13) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
Balancing design tradeoffs: cost performance schedule and risk	2	0	3	3	3	0	0	0	0	0	3
Writing project proposals	0	0	0	3	3	0	3	0	0	0	3
Team-project organization and management (including time lines)	0	0	2	3	0	0	3	0	0	0	3
Requirements capture and analysis	2	0	3	3	3	0	0	0	0	0	3
Design and architecture	2	0	0	3	3	0	0	0	0	0	3
Prototyping (possibly via simulation)	3	3	0	3	3	0	0	0	0	0	3
Vertification / Validation	3	3	0	3	0	0	0	0	0	0	3
Writing and presenting final reports	0	0	0	3	0	0	3	0	0	0	3
Engineering professionalism and responsibility	0	0	0	0	0	3	0	0	0	1	3
Engineering careers and the modern world	0	0	0	0	0	3	0	3	3	3	3
Contemporary social and ethical issues in computer science	0	0	0	0	0	0	0	0	0	3	0

Prepared by, and date of preparation:
Christian Shelton – 06/15/06

Instructor's Manual for the Coverage of Ethical, Social, and Professional Responsibility in UCR's CS Curriculum

February 21, 2007

Abstract

This document details the instruction and testing in ethical, social, and professional-responsibility issues in UCR's CS curriculum. It begins with a short description of what is necessary and how to coordinate teaching among the courses. It concludes with suggestions on how to organize the material to help instructors get started.

1 Requirements

Issues of social, ethical and professional responsibility must be taught in the following four courses.

- CS 152 (Compiler Design)
- CS 153 (Design of Operating Systems)
- CS 161 (Design and Architecture of Computer Systems)
- CS 179 (Project in Computer Science)

Instructors of these courses should spend at least the equivalent of one class period (i.e., one hour) covering such topics and the students must be tested on their understanding either through *questions on the final exam* specifically designed for this topic or *a graded individual essay* due during the term. These per-item grades should appear in the course offering's gradebook.

1.1 Coordination

Each course should cover a different aspect of social, ethical, and professional responsibility; it is not useful to present students with the same material in four separate courses. We propose the following areas for each course:[1]

- CS 152 (Compilers): Computer Crime & Software Integrity
- CS 153 (OS): Privacy and Information Use
- CS 161 (Architecture): Property Rights
- CS 179 (Senior Project): Responsibility & Social Implications

Below, this document gives examples of each to help instructors generate classroom material and discussion topics.

Instructors are encouraged to make use of current events to drive the in-class discussion. The topics above have been assigned to classes based on their natural affinity with the courses' technical material. Those specific allocations should not be seen as rigid. If instructors find other topics that fit, they can change the suggested topic, provided that they coordinate with the current instructors for the other three courses and assure that the a reasonable range of topics and examples is covered.

Note: instructors in all courses are encouraged to make occasional digressions to discuss relevant contemporary social and ethical issues.

2 Ideas for Instruction

This section gives some ideas for material to be covered, examples that could be used (or have been used), and methods for testing comprehension for each of the areas listed above.

2.1 Testing Comprehension

It is not sufficient to merely discuss ethical and social issues in class. Testing is best done by asking the students to analyze a situation and provide an analysis. Essays are well-suited to this, but a short response question on an exam would also work. Alternatively (or perhaps in conjunction), having students report (and analyze) a current event is a good test of comprehension.

[1] These topics are taken loosely from the chapters of Deborah G. Johnson, Computer Ethics: second edition, Prentice Hall, 1994. It provides a reasonable guide to organizing a lecture and class discussion on any of these topics.

The criteria for judging such an essay are not straight-forward. Yet, the instructor should look for evidence that the student has thought about the situation from multiple points-of-view, reasoned about the competing interests, and come to a conclusion through analytic reasoning.

In CS 179 (Project in Computer Science), it is especially recommended that the instructor require an individual essay from each student on a social, ethical, or professional topic. It should be related (to the degree possible) to the student's project. Possible essays include analyzing

- the social or economic impact of the project,
- data privacy and security of the project,
- a specific moral or ethical dilemma resulting from working in teams,
- or a specific dilemma related to the implementation of the project.

For example, the instructor might assign an essay asking the students to evaluate a situation in which their manager directs them to stop working on their current project and implement a beta-tester online forum with censorship (say blocking messages that point out fatal flaws in the software).

2.2 Computer Crime & Software Integrity

Viruses, denial of service attacks, stolen data, and the like are pretty well known at this point. Some technical coverage or background might be helpful, but probably isn't necessary except to demonstrate the ease of creating security holes and the ease in exploiting them. Coverage of one or more of the following points would probably work well.

- **Hacker Ethics**: Hackers often contend that their actions are ethical. Some of the arguments include
  - Information should be free (and if it were, intellectual property and security would be unnecessary).
  - Breaking a system helps identify security problems.
  - (For some attacks) No harm was done.

  These arguments are not without some value and can be analyzed from a variety of positions.
- **Responsibility for Bugs**: If a bug (like a stack over problem) is found and exploited, who is responsible for it? Is the system designer responsible for the exploit? A quick survey of end-user agreements might be eye-opening for students.

In a compilers course, discussion of heavy language checking versus usability (like Ada vs. C) might be instructive. Verification (and the issues with specification authoring) is another possible subject.

Historic hacking accounts include the Robert Morris case and the Craig Neidorf case. The former is interesting because he tried to stop the worm once it got out of control. (It replicated 14-in-15 times instead of the desired 1-in-15 times using stack overflow in fingerd). The latter is interesting in that the "stolen" data was later revealed to be in the public domain.

2.3 Privacy and Information Use

File-system level privacy issues are common. However, a discussion of other privacy-level issues can naturally arise at the same time:

- **Quantitative Changes leading to Qualitative Differences:** Many data have always been publicly available. However, the ability to quickly collate and mine such data qualitatively changes the privacy associated with it: housing prices, buyer spending habits, e-mail, medical records, and workplace monitoring.

  Is there a hard and fast line between public and private data? Examples like zillow.com are current and germane to the students.

- **Data Spread:** There are no standard methods for securing data like medical or credit records. Once someone has the information, you cannot stop it from propagating further.

  Discussion of legal and technical methods for trying to control the spread of personal information. How to fix faulty information.

- **Privacy and Entities:** In UNIX-like file system, the notion of privacy is very individual-based (groups are a very small step away from this). More general databases or data records have more complex data relationships.

  Medical records exist between an individual and a doctor, but they also extend to the individual's family (to some degree) and to other doctors (to some degree). Technical, legal, and social solutions can be discussed.

A few general points can be brought up for any of the above discussions:

- A discussion of the "if an individual has nothing to hide, he or she should not worry" rebuttal. This is clearly flawed to some degree when we consider mistakes that enter databases.

- A discussion of the "an individual does not have to give out any information" (or can actively give out false information) rebuttal. Again, clearly not a complete argument.
- The ACM Code of Professional Conduct specifically states that members will address individuals' privacy, provide for the right kinds of security, and minimize the amount of information gathered.

2.4 Property Rights

Some basic legal definitions are probably helpful for students:

- A patent covers a method or process and protects the invention for 17 years. Applying for one is costly. Does not apply to scientific laws or mathematical formulas.
- A copyright covers an expression (but not the underlying idea) and applies for 70 years. It is automatic upon creation but can be applied for easily.
- A trade secret varies more in definition, but must be kept internal to the company in order to be a valid claim.
- A trademark guards a particular word, phrase, or logo.

Some possible topics of discussion include:

- **Patent war chests**: the goals and rationale behind companies patenting many things (and buying companies for their patents). How a typical patent war plays out (usually there is some contemporary court case in the news about this).
- **Software and patents and copyrights:** Why should (or should not) software be patentable. The patent process. Prior Art. Novelty.
- **Why** have protection for intellectual material? Does it encourage or discourage intellectual endeavors?
- **Software Piracy**: given that software can be copied without degrading the original, is anything actually stolen?
- **Free software**: "free" as in beer versus "free" as in speech. GNU software. Licenses.

2.5 Responsibility and Social Implications

Issues such as "quantitative changes leading to qualitative differences" and the ACM Code of Professional Conduct (from above) apply here as well. Revisiting

them (especially in the context of the students' projects) could be valuable. Additional possible topics include

- **Warranties:** Implied warranties, express warranties. Is software a product or service?
- **Online liability:** Who is liable for information on a message board? Who is responsible for tracking illegal online activity? Are they similar to prior models (newspapers)?
- **Workplace changes due to automation**: how do changes in automation affect society? The economics of automation. The social effects of automation.
- **Refusal to work on a project**: Does not working on a project absolve one's responsibility? Does not working change anything? Is "scientific progress" inevitable?
- **Autonomy**: Does the autonomy that computers provide strengthen or weaken societal bonds? Are online interactions as "deep" and socially meaningful as face-to-face ones? Has the relative cheapness of web publication improved information dissemination?
- **Access**: Do computers (and technologies in general) increase or decrease the disparity between "haves" and "have-nots"?