Appendix C: Course Exemplars

While the Body of Knowledge lists the topics and learning outcomes that should be included in undergraduate programs in Computer Science, there is a variety of ways in which these topics may be packaged into courses. Presently, we give a list of *course exemplars*, which provide examples of fielded courses from a variety of institutions that cover portions of the CS2013 Body of Knowledge in different ways. These examplars are not meant to be prescriptive with respect to curricular design—they are not meant to define a standard curriculum for all insitutions. Rather these course examplars are provided to give educators guidance on different ways that that portions of the Body of Knowledge may be organized into courses and to spur innovative thinking in future course design. Table A1 below provides a list of the course examplars in the order in which they appear in this Appendix. Table A2 provides a list of the same course examplars, organized by the Knowledge Area from the Body of Knowledge that they most significantly cover. As can be seen from these exemplars, a course often includes material from multiple Knowledge Areas and in some cases multiple courses may be used to cover all the material from one Knowledge Area.

Table A1: Courses by Course Title

Title	Institution	Major KAs	Page
582219 Operating Systems	Univ. of Helsinki	OS	224
CS188: Artificial Intelligence	UC Berkeley	IS	226
CIS133J: Java Programming I	Portland Community College	SDF, PL	228
CMSC 471: Introduction to Artificial Intelligence	U. Maryland Baltimore County	IS	231
COS126: General Computer Science	Princeton University	SDF, AL, AR	233
COS 226: Algorithms and Data Structures	Princeton University	AL	237
COSC/Math 201: Modeling and Simulation for the Sciences	Wofford College	CN	240
CPSC 3380 Operating Systems	University of Ark. Little Rock	OS, PD	244
CS 150 Digital Logic Design	UC Berkeley	AR	246
CS 152 Computer Engineering	UC Berkeley	AR	248
CS 2200 Computer Systems and Networks	Georgia Tech	SF	250
CS 420: Operating Systems	Embry Riddle Aeronautical University	OS, PD	254
CS 522 Introduction to Computer Architecture	U. Wisconsin-Madison	AR	256
CS 61c: Great Ideas in Computer Architecture	UC Berkeley	SF	259
CS 662: Artificial Intelligence Programming	U. San Francisco	IS	261
CS1101: Introduction to Program Design	Worcester Polytechnic Institute	SDF, PL	263
CS175 Computer Graphics	Harvard	GV, SE	266

CS371: Computer Graphics	Williams College	GV, SE	269
CS453: Introduction to Compilers	Colorado State University	PL, SE	272
CS5: Introduction to Computer Science	Harvey Mudd College	SDF, AL, AR	275
CSC 131: Principles of Programming Languages	Pomona College	PL	278
CSC 453: Translators and Systems Software	Univ. Arizona, Tucson	PL	281
Overview of Multi-Paradigm 3-Course CS Introduction	Grinnell College		283
CSC151: Functional problem solving	Grinnell College	SDF, PL, AR	285
CSC161: Imperative Problem Solving and Data Structures	Grinnell College	SDF, PL	287
CSC207: Algorithms and Object-Oriented Design	Grinnell College	SDF, AL, PL	289
Overview of 2-Course Introduction Sequence	Creighton University		292
CSC221: Introduction to Programming	Creighton University	SDF, PL	293
CSC222: Object-Oriented Programming	Creighton University	SDF, PL, AL	295
CSCI 0190: Accelerated Introduction to Computer Science	Brown Univ.	PL, SDF, SE, AL	297
CSCI 140: Algorithms	Pomona College	AL	299
CSCI 1730: Introduction to Programming Languages	Brown Univ.	PL	302
CSCI 256: Algorithm Design and Analysis	Williams College	AL	304
CSCI 334: Principles of Programming Languages	Williams College	PL, PD	307
CSCI 432 Operating Systems	Williams College	OS	310
CSCI 434T: Compiler Design	Williams College	PL	313
CSE 333 System Programming	U. Washington	SF, OS, PL	316
CSE 332: Data Abstractions	U. Washington	AL, PD	319
Discrete Mathematics	Union County College	DS	322
Discrete Structures 1	Portland Community College	DS	325
Discrete Structures 2	Portland Community College	DS, AL	328
Ethics & the Information Age (CSI 194)	Anne Arundel Community College	SP	331
Ethics in Technology (IFSM 304)	University of Maryland, University College	SP	334
Human Aspects of Computer Science	University of York, UK	HCI	337
Human Computer Interaction	University of Kent, UK	HCI	339
Introduction to Artificial Intelligence	Case Western Reserve Univ.	IS	341
Introduction to Artificial Intelligence	U. Hartford	IS	344
Introduction to Parallel Programming	Nizhni Novgorod State University	PD	347
Issues in Computing	Saint Xavier University	SP	349
Languages and Compilers	Utrect University	PL, AL	351
Professional Development Seminar	Northwest Missouri State University	SP	353
Programming Languages	U. Washington	PL	356
Programming Languages and Techniques I	Univ. of Penn.	PL, SDF	359
SE 2890 Software Engineering Practices	Milwaukee School of Engineering	SE	362
Software Engineering Practices	Embry Riddle Aeronautical University	SE	364
Technology, Ethics, and Global Society (CS 262)	Miami University (Oxford, OH)	SP, HCI, GV	368
Topics in Compiler Construction	Rice	PL, AL	371
CS103/CS109: Mathematical Foundations of CS /Probability for Computer Scientists	Stanford Univ.	DS, AL	374

Table A2: Exemplars by Knowledge Area

NOTE: Courses listed below in parentheses have a secondary emphasis in this area.

KA	Course		Page
AL	Pomona College	CSCI 140: Algorithms	299
	Princeton University	COS 226: Algorithms and Data Structures	237
	Williams College	CSCI 256: Algorithm Design and Analysis	304
	U. Washington	CSE332: Data Abstractions	319
	Grinnell College	CSC207: Algorithms and Object-Oriented Design	289
	(Harvey Mudd College	CS5: Intro to Computer Science)	275
	(Portland Community College	Discrete Structures 2)	328
	(Utrect	Languages and Compilers)	351
	(Princeton University	COS126: General Computer Science)	233
AR	U. Wisconsin-Madison:	CS522: Intro to Computer Architecture	256
	UC Berkeley:	CS150: Digital Logic Design	246
	UC Berkeley:	CS152: Computer Engineering	248
	(Harvey Mudd College:	CS5: Intro to Computer Science)	275
CN	Wofford College	COSC/Math 201: Modeling and Simulation	240
DS	Union County College	Discrete Mathematics	322
	Stanford Univ.	CS103/CS109: Mathematical Foundations of CS and Probability for CS	374
	Portland Community College	Discrete Structures 1	325
	Portland Community College	Discrete Structures 2	328
GV	Harvard	CS175:Computer Graphics	266
	Williams College	CS371: Computer Graphics	269
	(Miami University (Oxford, OH)	CS262:Technology, Ethics, and Global Society)	368
HCI	University of York, UK	Human Aspects of Computer Science	337
	(University of Kent, UK	Human Computer Interaction)	339
	(Miami University (Oxford, OH)	Technology, Ethics, and Global Society (CS 262))	368
IAS	*forthcoming*		
IM	*forthcoming*		
IS	U. San Francisco	Artificial Intelligence Programming	261
	U. Maryland, Baltimore County	Introduction to Artificial Intelligence	231
	Case Western Reserve Univ.	Artificial Intelligence	341
	UC Berkeley	CS188: Artificial Intelligence	226
	Univ. Hartford	Artificial Intelligence	344
NC	*forthcoming*		
OS	Williams College	CSCI 432: Operating Systems	310
	University of Ark. Little Rock	CPSC 3380: Operating Systems	244
	Embry Riddle Aeronautical Univ.	CS 420: Operating Systems	254
	Univ. of Helsinki	582219 Operating Systems	224
PBD	*forthcoming*		
PD	Nizhni Novgorod State University	Introduction to Parallel Programming	347
	(U. Washington	CSE332: Data Abstractions)	319
	(University of Ark. Little Rock	CPSC 3380: Operating Systems)	244
	(Embry Riddle Aeronautical Univ.	CS 420: Operating Systems)	254
	(Williams College	CSCI 334: Principles of Programming Languages)	307
PL	*Compilers*		
	Colorado State University	CS 453: Introduction to Compilers	272
	Univ. Arizona, Tucson	CSC 453: Translators and Systems Software	281
	Williams College	CSCI 434T: Compiler Design	313

	Utrect	Languages and Compilers	351
	Rice	Topics in Compiler Construction	371
	Programming Languages		
	Pomona College	CS 131: Principles of Programming Languages	278
	Brown Univ.	CSCI 1730: Introduction to Programming	302
	U. Washington	Programming Languages	356
	Williams College	CSCI 334 Principles of Programming	307
	Univ. of Penn.	Programming Languages and Techniques I	359
	(Brown Univ.	CSCI 0190: Accelerated Intro. to Computer Science)	297
	(Portland Community College	CIS 133J: Java Programming I)	228
	(Worchester Polytechnic Inst.	CS1101: Introduction to Program Design)	263
SDF	***Also see Introductory Sequences (at end of table)***		283, 292
	Portland Community College	CIS133J: Java Programming I	228
	Harvey Mudd College	CS5: Introduction to Computer Science	275
	Worchester Polytechnic Inst.	CS1101: Introduction to Program Design	263
	(Univ. of Penn.	Programming Languages and Techniques I)	359
	(Princeton University	COS126: General Computer Science)	233
	(Brown Univ.	CSCI 0190: Accelerated Intro. to Computer Science)	297
SE	Embry Riddle Aeronautical Univ.	Software Engineering Practices	364
	Milwaukee School of Engineering	SE 2890:Software Engineering Practices	362
	(Colorado State University	CS453: Introduction to Compilers)	272
	(Harvard	CS175 Computer Graphics)	266
	(Williams College	CS371: Computer Graphics)	269
	(Brown Univ.	CSCI 0190: Accelerated Intro. to Computer Science)	297
SF	Georgia Tech	CS 2200: Computer Systems and Networks	250
	UC Berkeley	CS 61c: Great Ideas in Computer Architecture	259
	U. Washington	CSE 333: System Programming	316
SP	Univ. of Maryland, Univ. College	Ethics in Technology (IFSM 304)	334
	Saint Xavier University	Issues in Computing	349
	Anne Arundel Community College	Ethics & the Information Age (CSI 194)	331
	Miami University (Oxford, OH)	Technology, Ethics, and Global Society	368
	Northwest Missouri State Univ.	Professional Development Seminar	353
Introductory Sequences	Creighton University		292
	CSC221: Introduction to Programming		
	CSC222: Object-Oriented Programming		
	Grinnell College		283
	CSC207: Algorithms and Object-Oriented Design		
	CSC161: Imperative Problem Solving and Data Structures		
	CSC151: Functional problem solving		

SE-2890 Software Engineering Practices

Milwaukee School of Engineering
Walter Schilling
schilling@msoe.edu

Knowledge Areas that contain topics and learning outcomes covered in the course

Knowledge Area	**Total Hours of Coverage**
Software Engineering (SE)	*29*

Where does the course fit in your curriculum?

Second-year course for computer engineers covering SE fundamentals.

Prerequisites: one year of Java software development including use and simple analysis of data structures. Students have also had two one-quarter courses in 8-bit microprocessor development with assembly language and C.

What is covered in the course?

Week 1 - Introduction to software engineering practices
Week 2 - Requirements and Use Cases
Week 3 - Software Reviews, Version Control, and Configuration Management
Week 4/5 - Design: Object domain analysis, associations, behavior
Week 6 - Design and Design Patterns
Week 7 - Java Review (almost a year since last use)
Week 8/9 - Code reviews and software testing
Week 10 - Applications to embedded systems

What is the format of the course?

One-quarter (10-week), two one-hour lectures and one two-hour closed (instructor directed) lab per week.

How are students assessed?

Midterm and final exams, two individual lab projects and on 8-week team development project.

Course textbooks and materials

Gary McGraw, Real Time UML: Third Edition
Advances in the UML for Real-Time Systems Bruce Powel Douglass, Addison- Wesley, 2004.

Why do you teach the course this way?

The major goal is to prepare computer engineering students (not SE majors) to work in a small team on a small project, and to gain an introduction to software engineering practices.

Body of Knowledge coverage

KA	**Knowledge Unit**	**Topics Covered**	**Hours**
SE	*Software Processes*		*4*
SE	Software Project Management		2
SE	Tools and Environments		3
SE	Requirements Engineering		6

SE	Software Design		10
SE	Software Verification & Validation		4

Software Engineering Practices

Embry Riddle Aeronautical University, Daytona Beach, Florida
Salamah Salamah
salamahs@erau.edu

Knowledge Areas that contain topics and learning outcomes covered in the course

Knowledge Area	**Total Hours of Coverage**
Software Engineering	42

Where does the course fit in your curriculum?

This is a junior level course required for students majoring in software engineering, computer engineering, or computer science. The course is also required by those students seeking a minor in computer science.

The course has a an introductory to computer science course as a prerequisite.

The typical population of students in the course is between 30-35 students.

What is covered in the course?

Typical outline of course topics includes:

- Introduction to Software Engineering
- Models of Software Process
- Project Planning and Organization
- Software Requirements and Specifications
- Software Design Techniques
- Software Quality Assurance
- Software Testing
- Software Tools and Environments

What is the format of the course?

The course meets twice a week for two hours each day. The course is a mixture of lecture (about 1.5 hours a week) and group project work. The course is structured around the project development where the students are constantly producing artefacts related to software development life cycle.

How are students assessed?

Students are assessed through multiple means. This includes

- Individual programming assignments (about 3 a semester)
- In class quizzes
- Homework assignments
- Two midterms
- Semester long team project

Students peer evaluation is also part of the assessment process.

Course textbooks and materials

Watts Humphrey's Introduction to the Team Software Process is the primary book for the course, but this is also complemented with multiple reading assignments including journals and other book chapters.

Why do you teach the course this way?

The course is taught as a mini capstone course. It has been taught this way for the last 7 years at least. Students' comments indicate that the course is challenging in the sense that it drives them away from the perceived notion that software engineering is mostly about programming. Course is only reviewed annually as part of the department assessment and accreditation process.

I believe teaching the course based on a semester project is the easiest way to force students to apply the concepts and get familiar with the artefacts associated with a typical software development process.

Body of Knowledge coverage

KA[2]	Knowledge Unit	Topics Covered	Hours
SP	System Level Consideration	Relation of software engineering to Systems Engineering Software systems' use in different domains Outcome: Core-Tier1 # 1	*1*
SP	Software Process Models	Waterfall model Incremental model Prototyping V model Agile methodology Outcome: Core-Tier1 # 2 Outcome: Core-Tier1 # 3 Outcome: Core-Tier2 # 1 Outcome: Core-Tier2 # 2	2
SP	Software Quality Concepts	Outcome: Elective # 1 Outcome: Elective # 4 Outcome: Elective # 6 Outcome: Elective # 7	4
PM	Team Participation	Outcome: Core-Tier2 # 7 Outcome: Core-Tier2 # 8 Outcome: Core-Tier2 # 9 Outcome: Core-Tier2 # 11	2
PM	Effort Estimation	Outcome: Core-Tier2 # 12	2
PM	Team Management	Outcome: Elective # 2 Outcome: Elective # 4 Outcome: Elective # 5	1
PM	Project Management	Outcome: Elective # 6 Outcome: Elective # 7	2
RE	Fundamentals of software requirements elicitation and modelling	Outcome: Core-Tier1 # 1	1
RE	Properties of requirements	Outcome: Core-Tier2 # 1	1

[2] Abbreviation of Knowledge areas is available in the table at the end of the document.

RE	Software Requirement Elicitation	Outcome: Core-Tier2 # 2	1
RE	Describing functional Requirements using use cases	Outcome: Core-Tier2 # 2	1
RE	Non-Functional Requirements	Outcome: Core-Tier2 # 4	1
RE	Requirements Specifications	Outcome: Elective # 1 Outcome: Elective # 2	2
RE	Requirements validation	Outcome: Elective # 5	1
RE	Requirements Tracing	Outcome: Elective # 5	1
SD	Overview of Design Paradigms	Outcome: Core-Tier1 # 1	1
SD	Systems Design Principles	Outcome: Core-Tier1 # 2 Outcome: Core-Tier1 # 3	1
SD	Design Paradigms (OO analysis)	Outcome: Core-Tier2 # 1	1
SD	Measurement and analysis of design qualities	Outcome: Elective # 3	1
SC	Coding Standards	Outcome: Core-Tier2 # 4	2
SC	Integration strategies	Outcome: Core-Tier2 # 5	1
VV	V&V Concepts	Outcome: Core-Tier2 # 1	1
VV	Inspections, Reviews and Audits	Outcome: Core-Tier2 # 3	3
VV	Testing Fundamentals	Outcome: Core-Tier2 # 4 Outcome: Core-Tier2 # 5	2
VV	Defect Tracking	Outcome: Core-Tier2 # 6	1
VV	Static and Dynamic Testing	Outcome: Elective # 1	2
VV	Test Driven Development	Test Driven Development Programming Assignment No available outcome	1
SE	Characteristics of maintainable software	Lecture on software maintenance and the different types of maintenance No available outcome	1

SE	Reengineering Systems	Lecture on reverse engineering No available outcome	1
FM	Role of formal specifications in software development cycle	Outcome 1 Outcome 2 Outcome 3	2
SR	None		0

Additional topics

Ethics

Other comments

Knowledge Areas Abbreviations

Knowledge Area	Acronym
Software Process	SP
Software Project Management	PM
Tools and Environment	TE
Requirements Engineering	RE
Software Design	SD
Software Construction	SC
Software Validation and Verification	VV
Software Evolution	SE
Formal Methods	FM
Software Reliability	SR