SOFTWARE ENGINEERING

Lesson Plan and Student Workbook

Table of Contents

Course Syllabus 2
Course Schedule 3
Module 1. Software Engineering Overview 4
Module 2. Software Project Management 5
Module 3. Software Requirements 6
Module 4. Software Design 7
Module 5. Programming Language Issues 8
Module 6. Software Testing 9
Module 7. Delivery and Maintenance 10

The following pages give the reading assignments, programming assignments, reading and study guides, and other activities for this course.

Course Syllabus

Class time and location: MWF 2:00-2:50, Rhodes
Instructor: Prof. Hal Carter, Rhodes 814
Teaching assistant:

Course Summary and Objectives:

The purpose of this course is to give the student both theory and methods for conducting all phases of software product development and support. Particular emphasis is given to project management, specifications analysis, design techniques, coding styles, quality assurance, testing and maintenance of software. The student is provided with a number of quantitative approaches to each of the software development phases which, when applied in software engineering activities, increases the successful completion of high-quality software products.

Prerequisites:

ECE 226 - Intro to Programming; ECE 328 - Data Structures and Algorithms

Co-Requisites:

ECE 495 - Software Engineering Lab

Required Texts:

Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 3rd Edition, McGraw-Hill, 1992.

Grading:

The grades for this course are based upon the following weighted basis:

Item	% grade
Homework	15
Quizzes	15
MidtermExam	35
Final Exam	35
Total	100

Homework:

All homework is due at the beginning of class on the Monday immediately following the day the homework is assigned. Late home work is penalized 10% of the earned grade.

Course Schedule

Wk	Day	Date	Description	Reading[1]	HW[1]
1			**** Module 1: Software Engineering Overview****		
	M	Jan 6	Introduction to Course		
	W	Jan 8	Intro to software engineering	Ch 1	
			**** Module 2: Software Project Management ****		
	F	Jan 10	Software metrics	Ch 2	2.4
2	M	Jan 13	Project planning	Ch 3	3.8[2]
	W	Jan 15	Project planning	Ch 4	4.13, 4.15
			**** Module 3: Software Requirements ****		
	F	Jan 17	Computer systems engineering	Ch 5	
3	M	Jan 20	HOLIDAY		
	W	Jan 22	Requirements Analysis	Ch 6	6.9
	F	Jan 24	Requirements analysis	Ch 6	6.10
4	M	Jan 27	Structured analysis	Ch 7	7.11
	W	Jan 29	Structured analysis (cont.)	Ch 7	
	F	Jan 31	Object oriented analysis	Ch 8	8.4
5	M	Feb 3	Object oriented analysis (cont.)	Ch 8	8.5
	W	Feb 5	Alternative analysis techniques	Ch 9	
	F	Feb 7	Formal analysis models	Ch 9	9.6
6	M	Feb 10	Midterm Exam		
			**** Module 4: Software Design ****		
	W	Feb 12	Software design fundamentals	Ch 10	10.1
	F	Feb 14	Software design fundamentals (cont.)	Ch 10	10.17
7	M	Feb 17	Data flow-oriented design	Ch 11	11.10
	W	Feb 19	Object-oriented design	Ch 12	
	F	Feb 21	Object-oriented design (cont.)	Ch 12	12.10
8	M	Feb 24	Data-oriented design	Ch 13	13.4
	W	Feb 26	User interface design	Ch 14	
	F	Feb 28	Real-time design	Ch 15	15.8
9			**** Module 5: Prog. Language Issues ****		
	M	Mar 1	Languages and Coding	Ch 16	16.10
			**** Module 6: Software Testing ****		
	W	Mar 3	Software integrity	Ch 17	17.15
	F	Mar 5	Software testing techniques	Ch 18	18.1
10	M	Mar 8	Software testing procedures	Ch 19	19.4
			**** Module 7: Delivery and Maintenance ****		
	W	Mar 10	Software maintenance	Ch 20	
	F	Mar 12	Software configuration management	Ch 21	
		Mar 9-13	Final Exam		

[1] Course Text
[2] Use project in Software Engineering Lab. Use results of this exercise in your SPP in EE 495

Module 1. Software Engineering Overview

Module Objectives

The objectives of this module are:

- Present and discuss the idea that software is more than just code -- *engineered* software is composed of *controlled configuration items* which include documents, data, and code.
- Present and discuss the history of software development, including its evolution into a business.
- Present and discuss many of the problems with doing software development, particularly when there is more than one person involved.
- Present and discuss many myths about software development and explain why some of these myths are fallacies.
- Present and discuss several different software engineering paradigms, showing different methods for developing engineered software:
    1. Classic "waterfall" method
    2. Rapid prototyping
    3. Spiral method
    4. Fourth generation method
    5. Combinations of the above
- Present and discuss some of the technologies used in the support of software engineering:
    1. *Computer-Aided Software Engineering* (CASE)
    2. Ada programming language

Topic areas presented in this lecture are:

- Engineered Software
- Controlled Configuration Item
- History of Software Development
- Software as a Business Opportunity
- Problems in Software Development
- Myths about Software Development
- Approaches to Software Development
- Engineered Software Development Models
- Software Engineering Technology

Reading Assignment

Pressman, Chapter 1

Study Guide

✓ What are the first four generations of computer software?

✓ Itemize at least four questions raised by industry regarding the development of software into an effective business.

✓ How does the failure curve for software differ from the failure curve for hardware and why?

✓ Itemize the elements of the classic "waterfall" software lifecycle.

✓ Explain the spiral model for software development.

✓ What are the three main phases of software engineering? Describe them in detail.

Module 2. Software Project Management

Module Objectives

The objectives of this module are:

- Present and discuss the six items a project leader must determine in any software project planning.
- Identify the tasks required duriing the planning process.
- Explain and motivate the need for software metrics. including how to collect software metrics.
- Explain the two primary types of size metrics: lines of code (LOC) and function points.
- Identify the resources necessary to carry out a major software development.
- Show how software metrics are used to identify cost and schedule estimates.
- Explain and demonstrate the use of the COCOMO estimation model.
- Identify the key elements of software project planning.
- Explain and demonstrate risk management, monitoring, and tracking.
- Describe approaches to project scheduling.
- Describe approaches to tracking software development.
- Define software acquisition and show an approach for make-or-buy decisions.
- Describe software re-engineering and when it is to be used.
- Describe organizational planning.
- Present the basic elements of the SOftware Project Plan (SPP).

Topic areas presented in this module are:

- Software metrics
- Software cost and timing estimation
- Software project planning

Reading Assignment

Pressman, Chapters 2, 3, and 4

Study Guide

- ✓ Why are software metrics so important to software development?
- ✓ Make list of at least five major metrics and describe how to obtain them.
- ✓ A lot of controversy surrounds the acquisition and use of software metrics. Why?
- ✓ What is the relationship between LOC metrics and feature point metrics?
- ✓ What are the major determinants of software quality?
- ✓ Define software integrity and useability. Can you quantify your definitions?
- ✓ Using the spreadsheet data collection model, how would you apply it to estimate schedules and costs for a project?
- ✓ Explain the differences between a "compositional" vesus a "decompositional" approach to software project estimation.
- ✓ Using data of your choice, estimate the schedule and costs for software development using the COCOMO approach.
- ✓ Outline and describe each section of the software project plan.

Module 3. Software Requirements

Module Objectives

The objectives of this module are:

- Define the role and necessity of software requirements analysis
- Explain each of the the four basic priciple of requirements analysis
- Describe the behavioral, logical, and physical views of software requirements
- Explain software prototyping and how requirements are created within this paradigm
- Define software specifications and how they relate to software requirements
- Explain the eight basic principles of software specifications
- Define structured analysis
- Show how to use data flow diagrams in a realistic specification
- Describe at least one method for extending data flow diagrams to real-time systems
- Present a state transisition diagram approach to behavioral modeling of software
- Explain what a requirements dictionary is, how to create one, and where it fits in structured analysis
- Define object oriented analysis and contrast it with strutured analysis methods
- Expalin the elements of object oriented analysis and how to derive them for a specific example
- Describe object oriented analysis modeling and apply it to a specific example
- Describe data modeling and relate it to object oreinted analysis
- Explain how to use entity-relationship diagrams
- Explain at least on alternative data structure-oriented method
- Show an example of use of the Z specification method as a representation of formal specification methods

Topic areas presented in this module are:

- Fundamentsl of software requirements
- Structured analysis
- Object oriented analysis
- Alternative structured analysis techniques
- Format specifications

Reading Assignment

Pressman, Chapters 5, 6, 7, 8, and 9.

Study Guide

✓ What are the objectives of software requirements analysis?
✓ What deliverable are produced from the software requirements analysis activity?
✓ What are the fundamental activities performed during software requirements analysis regardless of the methodology used?
✓ Identify and explain the characteristics common to all software requirements analysis methodologies.
✓ In the context of software requirments analysis, what is meant by "noise?"
✓ Identify and explain the principles to be followed in creating a good software requirements specification.

Module 4. Software Design

Module Objectives

The objectives of this module are:

- Explain the fundamental software design concepts of abstraction, refinement, modularity, hierarchy, structure, behavior, and information hiding
- Demonstrate how one creates modular software design
- Identify sequential and parallel module types
- Explain functional independence
- Describe cohesion and relate it to functional strength of a module
- Describe coupling and relate it to the module independence
- Define data design
- Explain the process of data design as summarized by Wasserman
- Describe architectural design and procedural design and their relationship
- Demonstrate the flowcharting graphical design notation
- Describe the use of decision tables in software design
- Describe and apply a program design language (PDL)
- Define data flow-oriented design
- Define and contrast transform flow and transaction flow
- Demonstrate the use of transform analysis and transaction analysis
- Explain a set of design heuristics for further improving design structures
- Identify the additional requirements real-time systems impose on design
- Demonstrate the use of data flow diagrams used as queuing network models and relate them to teal-time design
- Describe the DARTS approach to real-time software design

Topic areas presented in this module are:

- Software dsign fundamentals
- Data flow-oriented design
- Object-oriented design
- Data-oriented design methods
- User interface design
- Real-time design

Reading Assignment

Pressman, Chapters 10, 11, and 15.

Study Guide

✓ What are the three primary aspects of a software design?
✓ How is the quality of a design assessed during the design process?
✓ What is meant by a) modularity, b) levels of abstraction, c) information hiding?
✓ What are at least two benefits of abstraction?
✓ What is an abstract data type?
✓ What is an abstract state machine?
✓ What is meant by the term "cohesion?"
✓ How many kinds of cohesion are there?
✓ What is meant by the term "coupling?"
✓ What kinds of coupling are there?
✓ Define the terms "fan in" and "fan out" in the context of software design.
✓ Name at least five concerns addressed in the design of real-time systems which are not found in non-real time systems.

Module 5. Programming Language Issues

Module Objectives

The objectives of this module are:

- Explain the psychology of programming
- Define the key charactersistics of programming languages
- Describe the essential fundamentals of programming languages
- Define the generational classes of languages
- Explain the major elements of coding style and relate them to quality of code

Topic areas presented in this module are:

- Procedural versus nonprocedural programming
- Goals of software engineering
- Language specific issues in programming
- Compiler issues
- Organizational issues
- Language selection

Reading Assignment

Pressman, Chapter 16

Study Guide

✓ Name three psychological characteristics of a programming language.
✓ Name eight criteria for the selection of a programming language.
✓ Name five basic ways in which one programming language may differ from another.
✓ Name four characterisitics of a programming language from an engineering viewpoint.
✓ Why are standards for coding style established?
✓ Name four aspects of coding style.

Module 6. Software Testing

Module Objectives

The objectives of this module are:

- Explain the eleven software quality factors defined by McCall and show their relationship
- Define the criteria for grading software quality as defined by McCall
- Use McCall's method to evaluate a real software program
- Define Software Quality Assurance (SQA)
- Describe the basic software reviews and how they contribute to software quality
- Explain the use of of the U.S. Air Force Systems Command design structure quality index (DSQI)
- Explain Halstead's software science and show its advantages and disadvantages
- Demonstrate the use of McCabe's complexity metric
- Explain statistical quality assurance and how it is used
- Define software reliability
- Describe the test information flow for software test
- Demonstrate the use of basis path testing including deriving test cases
- Explain control structure testing and show how it is used
- Describe data flow testing
- Demonstrate the use of black box testing
- Explain the black box testing concepts of equivalence partitioning, boundary value analysis, cause-effect graphing, and comparison testing
- Explain the difference between verification and validation
- Define the spiral view of a software testing strategy
- Explain unit testing
- Define integration testing
- Describe system testing

Topic areas presented in this module are:

- Quality assurance
- Black box testing
- White box testing
- Testing strategies

Reading Assignment

Pressman, Chapters 17, 18, and 19.

Study Guide

- ✓ What are the software quality metrics and grading metrics proposed by McCall, and how are they used to evaluate the quality of a software item?
- ✓ What are the major software reviews employed during software development?
- ✓ How is defect amplification conducted during reviews?
- ✓ What is the software maturity index (SMI), and how is it calculated?
- ✓ How does Halstead's method for determining software quality work?
- ✓ Can you appy McCabe's complexity metric to a software program of your choice?
- ✓ Where does cyclomatic complexity not work well in whitebox testing?
- ✓ How do condition testing, data flow testing, and loop testing work?
- ✓ Can you describe how each of the black box techniques work, and where they are applicable in the software test process?

Module 7. Delivery and Maintenance

Module Objectives

The objectives of this module are:

- Know the activites associated with softawre delivery
- Identify the primary characteristics of maintenance characteristics
- Explain the factors and measures of maintainability
- Comprehend the tasks associated with software maintenance
- Explain methods for controlling side effects
- Define conditions and rules for effectively maintianing old code
- Identify why reverse engineering is needed and how to perform to perform it
- Identify and explain software configuration items
- Describe the objects in software configuration
- Explain version and change control including methods for implementing them

Topic areas presented in this module are:

- Icebergs
- Software maintainability
- Software maintenance activities
- Software configuration
- Software configuration management
- The Software Configuration Management (SCM) process

Reading Assignment

Pressman, Chapters 21 and 22.

Study Guide

✓ What are the three primary viewpoints of software maintenance characteristics?
✓ What are the major poblems with software maintenance?
✓ What is the best way to maintain software?
✓ How does one most efficiently perform software reverse engineering and re-engineering?
✓ What are software configuration items?
✓ What are the primary issues regarding software configuration management?
✓ How is version and change control implemented?