

CS 305: Software Engineering
Marshall University, Fall 2006

Contents

1 Course Description	3
2 Instructor's Information	3
3 Course Topics at a Glance	3
4 Detailed Description of Course Topics	3
4.1 Software Engineering Overview Module	3
4.1.1 Introduction	4
4.1.2 Socio-technical Systems	4
4.1.3 Critical Systems	4
4.1.4 Software Processes	4
4.1.5 Project Management	4
4.2 Requirements Engineering Module	4
4.2.1 Software Requirements	4
4.2.2 Requirements Engineering Processes	5
4.2.3 System Models	5
4.2.4 Critical Systems Specification	5
4.2.5 Formal Specification	5
4.3 Software Design Module	5
4.3.1 Architectural Design	5
4.3.2 Distributed Systems Architecture	6
4.3.3 Application Architectures	6
4.3.4 Object-oriented Design	6
4.3.5 Real-time Software Design	6
4.3.6 User Interface Design	6
4.4 Management Module	6
4.4.1 Managing People	7
4.4.2 Software Cost Estimation	7
4.4.3 Quality Management	7
4.4.4 Process Improvement	7
4.4.5 Configuration Management	7

5	Course Assessment	8
5.1	Written Assignments	8
5.2	Team Project	8
5.3	Midterm Exam 1	8
5.4	Midterm Exam 2	8
5.5	Final Exam	9
6	Instructional Materials	9
7	WebCT Vista	9

1 Course Description

Software engineering is an *engineering discipline* which encompasses all aspects of software production from requirements elicitation, system specification, design, implementation, testing, deployment, and maintenance. In this course, you will learn and apply software engineering methods, best practices, and tools in developing a software system. Due to the broad nature of the subject area, this course will place emphasis on the breadth of the subject matter rather than an in-depth study of a few topics.

2 Instructor's Information

- Dr. V.N. Gudivada, Gullickson Hall Room 205A, Phone: 304-696-5452, Email: gudivada@marshall.edu.

Office hours: MWF 12.00 Noon - 2.00 PM. Other times by appointment.

3 Course Topics at a Glance

1. Overview of software engineering principles, systems engineering, critical systems, software processes, and project management.
2. Requirements engineering — requirements elicitation and specification, and system models.
3. Software design — architectural design, distributed systems, application architecture, object-oriented design, real-time software design, and user interface design.
4. Software development — rapid software development, software reuse, component-based software engineering, critical systems development, and software evolution.
5. Software verification and validation — inspections, static analysis, formal methods, and testing.
6. Management — team building and nurturing, software cost estimation, quality management, process management, and configuration management.
7. Emerging technologies — security engineering, service-oriented software engineering, and aspect-oriented software development.

4 Detailed Description of Course Topics

4.1 Software Engineering Overview Module

The goal of this module is to answer the following questions:

4.1.1 Introduction

1. What is *software engineering* and why is it important to my career as an Information Technology (IT) professional?
2. What *ethical and professional* considerations are important for my practice as an IT professional?

4.1.2 Socio-technical Systems

1. What is a *socio-technical system*?
2. What are *emergent system properties*?
3. What are the activities in *systems engineering process*?
4. How does *organizational context* of a system affect its design and use?
5. What are *legacy systems*? Why are they important to the operation of many businesses?

4.1.3 Critical Systems

1. What are *critical systems*?
2. What are the *dimensions* of system dependability?
3. How do you enhance *system dependability*?

4.1.4 Software Processes

1. What are *software processes* and *software process models*?
2. How does *Rational Unified Process* (RUP) integrates good software process practice?
3. How does *CASE technology* support software process activities?

4.1.5 Project Management

1. What are the roles and responsibilities of *software project managers*?
2. What are the compelling needs for *software project planning*?
3. What graphical representations are used to represent *project schedules*?
4. What is *risk management*? How is it used in software project management?

4.2 Requirements Engineering Module

The goal of this module is to answer the following questions:

4.2.1 Software Requirements

1. What are *user requirements* and *systems requirements*? Why these requirements should be written in two different ways?

2. What are the differences between *functional and non-functional* requirements?
3. How do you *document* software requirements?

4.2.2 Requirements Engineering Processes

1. What are the various techniques for *requirements elicitation* and *analysis*?
2. How do you *validate* requirements?
3. How do you *manage* requirements?

4.2.3 System Models

1. What is *scope creep*? Why is it important to establish the *boundaries of a system*?
2. How do you perform *behavioral modeling*? *data modeling*? *object modeling*?
3. How do you use *UML* for system modeling?

4.2.4 Critical Systems Specification

1. How do you identify *dependability requirements* for critical systems?
2. How do you generate *safety requirements* from system *risk analysis*?
3. How do you generate *security requirements* based on different types of threats to the system?
4. How do you specify *reliability requirements*?

4.2.5 Formal Specification

1. How does *formal requirements specification* techniques help discover problems in *system requirements*?
2. How do you use *algebraic techniques* for specifying *interfaces*?
3. How model-based formal techniques are used in specifying *system behavior*?

4.3 Software Design Module

The goal of this module is to answer the following questions:

4.3.1 Architectural Design

1. What *decisions* have to be made about the system architecture during the architectural design process?
2. What are the three complementary *architectural styles*?
3. How do you use a *reference architecture* to communicate architectural concepts and to assess system architectures?

4.3.2 Distributed Systems Architecture

1. What are the advantages and disadvantages of *distributed systems*?
2. What is *client-server* architectural model?
3. What is *distributed object system architecture*?
4. What is *CORBA* standard?
5. How are *peer-to-peer* and *service-oriented architectures* are used to implement interorganizational distributed systems?

4.3.3 Application Architectures

1. Explain *batch* and *transaction processing* as two fundamental architectural organization of business systems.
2. Discuss *event-processing* as a mechanism for structuring *command-driven software systems*.
3. Explain the structure and organization of *language-processing systems*.

4.3.4 Object-oriented Design

1. What are the *important activities* in a general object-oriented design process?
2. Explain different *models for documenting* an object-oriented design.
3. How do you use *UML* for representing design artifacts?

4.3.5 Real-time Software Design

1. How are real-time systems usually implemented as a set of *concurrent processes*?
2. Explain the role of a *real-time operating system*.
3. Describe a *generic process architecture* for monitoring and control systems, and data acquisition systems.

4.3.6 User Interface Design

1. Explain primary user *interface design principles*.
2. Explain different *user interaction styles* and list contexts in which each one is more appropriate.
3. What are the principal activities in the *user interface design process*?
4. List *usability attributes* and explain different approaches to interface evaluation.

4.4 Management Module

The goal of this module is to answer the following questions:

4.4.1 Managing People

1. What are the issues involved in *selecting and retaining staff* in a software development organization?
2. What factors influence *individual motivation* and discuss their *implication* for software project managers?
3. What are key issues of team working?
4. Describe the structure of People Capability Maturity Model.

4.4.2 Software Cost Estimation

1. Explain the fundamentals of *software costing*.
2. Discuss three *metrics* for software productivity assessment.
3. Why a range of techniques should be used when *estimating software costs* and *schedule*?
4. Explain the principles of the *COCOMO II model* for algorithmic cost estimation.

4.4.3 Quality Management

1. Explain *quality management process*.
2. Why *standards* are important in the quality management process?
3. What are *software metrics*? What are the differences between *predictor metrics* and *control metrics*?
4. How are *measurements* used in assessing some software quality attributes?
5. Discuss *current limitations* of software measurement.

4.4.4 Process Improvement

1. Why software *process improvement* is worthwhile?
2. Discuss how *software process factors* influence software quality and the productivity of developers.
3. Describe simple models for *software processes*.
4. What is *process capability* and *process maturity*?
5. Explain the general form of the *CMMI model* for process improvement.

4.4.5 Configuration Management

1. Why software *configuration management* is required for complex software systems?
2. Discuss *four fundamental* configuration management activities?
3. How are *CASE tools* used for configuration management?

5 Course Assessment

The course assessment components include: written assignments (10%), team project (20%), two midterm exams (50%), and a final (20%). Maximum possible score is 100. Course grade is awarded based on the following scheme:

<i>Score</i>	<i>Letter Grade</i>
≥ 90	A
$\geq 80 \ \& \ < 90$	B
$\geq 70 \ \& \ < 80$	C
$\geq 60 \ \& \ < 70$	D
< 60	F

5.1 Written Assignments

There will be several (about 25) small written assignments. Each one will have about 5 questions. Answers to these questions must be submitted via WebCT Vista. Written assignments will account for 10% of your course grade.

5.2 Team Project

There will be a semester-long team project. Typically a team consists of 2 to 3 students. Teams will work on requirements engineering (elicitation, validation, specification) and design for a software system. Details will be provided in a separate handout.

There will be a few intermediate deliverables and a final deliverable (which is an integration of intermediate deliverables with proper revision). The team projects accounts for 20% of your course grade.

5.3 Midterm Exam 1

This exam will test your ability to answer the questions raised in the following modules:

1. Software Engineering Overview (section 4.1)
2. Requirements Engineering (section 4.2)
3. Software Design (section 4.3)

Midterm exam 1 accounts for 25% of your course grade.

5.4 Midterm Exam 2

This exam will test your ability to answer the questions raised in the following modules:

1. Software Design (section ??)
2. Verification and Validation (section ??)

Midterm exam 2 accounts for 25% of your course grade.

5.5 Final Exam

This exam will test your ability to answer the questions raised in the following modules:

1. Management (section 4.4)
2. Emerging Technologies (section ??)

Final exam accounts for 20% of your course grade. Final exam will be held on 11 December 2006 from 10.15 AM to 12.15 PM in GH 211.

6 Instructional Materials

Required Textbook Ian Sommerville, *Software Engineering* (8th edition), ISBN: 0-321-31379-8, Pearson Education, 2007.

Additional Resources Course notes and other handouts will be available on WebCT Vista. URLs for additional resources will also be listed on the Vista.

7 WebCT Vista

It is important to visit WebCT Vista for up-to-date information about the course. It hosts all the course materials including assignments, handouts, lecture notes, and reading materials. Also, you will use the Vista for submitting your team project.