UNIVERSITY OF THE WESTERN CAPE

DEPARTMENT OF COMPUTER SCIENCE

COS 312

SOFTWARE ENGINEERING PART

Course Outline

Lecturing Schedule & Exercises

2012

Compiled by M. J. Norman, 2012

COS312 Software Engineering Part

This course is based on the recommendations of:

Software Engineering 2004
Curriculum Guidelines for Undergraduate
Degree Programs in Software Engineering
A Volume of the Computing Curricula Series
August 23, 2004
http://sites.computer.org/ccse/SE2004Volume.pdf

and in particular on SE201 from the above which says:

> **SE201 Introduction to Software Engineering**
> This is a central course, presenting the basic principles and concepts of software engineering and giving a firm foundation for many other courses described below. It gives broad coverage of the most important terminology and concepts in software engineering. Upon completing this course, students will be able to do basic modeling and design, particularly using UML. They will also have a basic understanding of requirements, software architecture, and testing.

Text Book: Software Engineering: A Practitioners Approach,
by Roger S. Pressman, 7th Ed, 2010 (Alternate Edition), McGraw-Hill
ISBN 978-007-126782-3 or MHID 007-126782-4

The Printed Student Package contains the following material:

- Course Outline / Lecturing Schedule
- Chapter Summaries
- Chapter Slides
- Chapter Exercises
- The following reading material:
  - Journal Paper: *Software Engineering: Process, Principles and Goals*
  - Unified Modeling Language (UML) Notes
  - Use Case Terminology

Chapter Summaries taken from the text book web site at:

http://paris.mcgraw-hill.com/sites/0073375977/information_center_view0/

And then selecting “Student Edition”.

There is also a Quiz available for each chapter – do this quiz as part of your learning/study programme.

Selected Chapter Exercises taken from the text book – these must be completed and written up in an answer book as the chapters are completed in the class – the tutor / lecturer assistant will collect these and mark them

Lecturing Schedule:

Week 1:
Chapter 1: Introduction to Software Engineering

Learning outcomes/key concepts:
Application domains; characteristics of software; framework activities; legacy software; practice; principles & goals; software engineering; software myths; software process; umbrella activities (configuration management, risk management, project management, quality assurance, formal technical review, measurement, reusability);

Selected text book exercises: 1.2; 1.3; 1.4; 1.5; 1.6; 1.10; 1.11

Week 1:
Chapter 2: Process Models

Learning outcomes/key concepts:
Component-based development; concurrent models; evolutionary process models; formal methods models; generic process model; incremental process models; prescriptive process models; process patterns; unified process;

Selected text book exercises: 2.1; 2.7; 2.8; 2.9; 2.10; 2.14; 2.16

Week 2:
Chapter 3: Agile Development

Learning outcomes/key concepts:
Agile software development; agile process; agile unified process; agility; extreme programming;

Selected text book exercises: 3.1; 3.2; 3.4; 3.7; 3.8; 3.9; 3.11

Week 3:
Chapter 4: Principles that Guide Practice

Learning outcomes/key concepts:
Core principles; principles that govern – requirements, coding, communication, deployment, design, modeling, planning, testing;

Selected text book exercises: 4.1; 4.2; 4.4; 4.5; 4.10; 4.13; 4.15

Week 4:
Chapter 5: Understanding Requirements

Learning outcomes/key concepts:
Analysis model; analysis patterns; collaboration; elaboration; elicitation; inception; negotiation; quality function deployment;

Selected text book exercises: 5.1; 5.2; 5.3; 5.6; 5.9; 5.11

Week 5:
Chapter 6: Requirements Modeling: Scenarios, Information & Analysis Classes

Learning outcomes/key concepts:
Activity diagram; analysis classes; associations; class-based modeling; data modeling; domain analysis; grammatical parse; requirements modeling; UML models; use cases;

Selected text book exercises: 6.1; 6.2; 6.3; 6.4; 6.5; 6.10

Week 6:
Chapter 8: Design Concepts

Learning outcomes/key concepts:
Abstraction; architecture; aspects; cohesion; data design; design process; functional independence; good design; information hiding;

Selected text book exercises: 8.1; 8.2; 8.3; 8.6; 8.7; 8.10; 8.11

Week 7:
Chapter 17: Software Testing Strategies

Learning outcomes/key concepts:
Alpha test; beta test; class testing; configuration review; debugging; deployment testing; acceptance testing; stress testing; performance testing;

Selected text book exercises: 17.1; 17.3; 17.4; 17.7; 17.8

Tutorials & Practical Assignments/Projects

Introduction and application use of a UML modeling plus exercises / assignments will be handed out at the practical session.

Week 1: Introduction to NetBeans – Tutorial and Case Study
Week 2: Designing Use-Cases – Tutorial and Case Study
Week 3: Designing Classes and Generating Java Code – Tutorial and Case Study
Weeks 4 – 7: Software Engineering project – designing and implementing an application in NetBeans using UML. The following will be assessed: your Use case diagram, class diagrams, code and you need to demonstrate the application.

Assessment

Continuous assessments (written, practical) concluding with a final group project assignment and presentation integrating the various components (i.e. software engineering, databases and HCI) of CSC312.

*** end ***