"Computer Science \ Software Engineering"

Second Class ***Lecture 1*** ***Dr. Shayma Mustafa***

Course Description

This course is an introduction to software engineering and teaching the student a software development stages from the beginning of building the software until the software used.

Subject Structure: 2 Theoretical hours + 2 practical hours

Textbook and Reference

- Software Engineering Apratitioner's Approach - Roger S. Pressman
- Software Engineering – Ian Sommerville

Grading the course:

- 50% Final exam.
- 20% Mid term exam
- 10% Quizzes
- 15 % for practical work depends on the lab exercises.
- 5 % Report

Course Goals:

1. To understand the basic steps of developing a large software project.
2. To be able work in a team on a large software project.
3. To be able to effectively analyze a programming problem.
4. To be able to create class, object, use case, interaction, sequence, and state machine diagrams in UML notation.
5. To be able to effectively design a solution to a programming problem.
6. To be able to assess potential sources of risk of failure for a large software project.

Required Tools/Software

- Enterprise Architect ENTERPRISE ARCHITECT

What is software?

- Computer programs and associated documentation
- Software products may be developed for a particular customer or may be developed for a general market
- Software products may be:
  - Generic - developed to be sold to a range of different customers
  - Custom - developed for a single customer according to their specification

What is a software Engineering?

- Software engineering is an engineering discipline which is concerned with all aspects of software production
- Software engineers should adopt a systematic and organised approach to their work and use appropriate tools and techniques depending on the problem to be solved, the development constraints and the resources available

Why study Software Engineering?

- Building software without discipline is crazy
- Software is critical to society
- Building a large complete software project is hard
- There is a perceived crisis in our ability to build software

What is the difference between software engineering and computer science?

- Computer science is concerned with theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.
- Computer science theories are still insufficient to act as a complete underpinning for software engineering (unlike e.g. physics and electrical engineering).

What is the difference between software engineering and system engineering?

- Software engineering is part of System engineering which concerned with all aspects of computer-based systems development including hardware, software and process engineering.
- System engineers are involved in system specification, architectural design, integration and deployment.

What is a software process?

- A set of activities whose goal is the development or evolution of a software system.
- Generic activities in all software processes are:
  - **Requirements Specification**: what the system should do and its development constraints.
  - **Design**: set of decisions on how the software system is going to meet its specification.
  - **Implementation**: actual production of the software system.
  - **Validation**: making sure that the implemented software is meeting its specification.
  - **Evolution**: changing the software in response to changing demands.

What are the costs in software engineering?

Roughly 60% of costs are development costs, 40% are testing costs. For custom software, evolution costs often exceed development costs.

Costs vary depending on the type of system being developed and the requirements of system attributes such as performance and system reliability, distribution of costs depends on the development model that is used.

What are the attributes of good software?

The software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable.

- Maintainability - Software must evolve to meet changing needs
- Dependability - reliable, safe, robust, secure
- Efficiency - in use of system resources
- Usability - must be usable by the users for which it was designed

The factors that affect software productivity:

1. The size of the team and their experience.
2. The personality of the team members.
3. The complexity of the problem.
4. If there are changes in requirements or design during the development cycle.
5. The SW design techniques that are applied and the review process that is used.
6. The implementation language, the software development tools, the computer H\W, and other S\W resources those are available.
7. If the S\W under development has reliability or performance requirements.