Course Title: Software Engineering
Course Code: SE-207

Course Teacher: Engr. Sidra,
Lecturer
Dept. of Software Engineering

Course Plan

WEEK No.	TOPICS	List of Material to be Uploaded	Assignment/Quiz Plan
0.	Introduction to Software Engineering	Lectures, Example Case Studies	
1.	Perspective Process Models	Lectures	
2.	Specialized Process Models	Lectures, Case Study	
3.	Agile Software Development	Lectures	Assignment # 1
4.	Requirement Engineering	Lectures	
5.	Design concepts,	Lectures	Quiz # 1
6.	System Modeling	Lectures	
7.	Architectural Design	Lectures	Assignment #2
8.	Testing Strategies	Lectures	
9.	Software Testing Fundamentals, Product Metrics	Lectures	
10.	Project Management	Lectures, Case Study	Quiz # 2
11.	Process and Project Metrics, Estimations for Software Projects	Lectures	
12.	Risk Management	Lectures,	Assignment # 3
13.	Quality Management	Lectures, Case Study	
14.	Advance Software Engineering	Lectures	
	Total per Semester:	45	

Sessional Criteria

Assessment Type	Marks	Schedule (Week No.)
Assignments	10	3,7,12
Quiz	10	5,10
Midterm	20	8
Total Sessional Marks	**40**	

- Reference Books:
  - “Software engineering”, Ian Sommerville, Addison Wesley, 10th Edition, 2015
  - “Software Engineering: A Practitioner’s Approach”, Roger S. Pressman, McGraw-Hill, 8 th Edition, 2014.

Introduction to Software Engineering

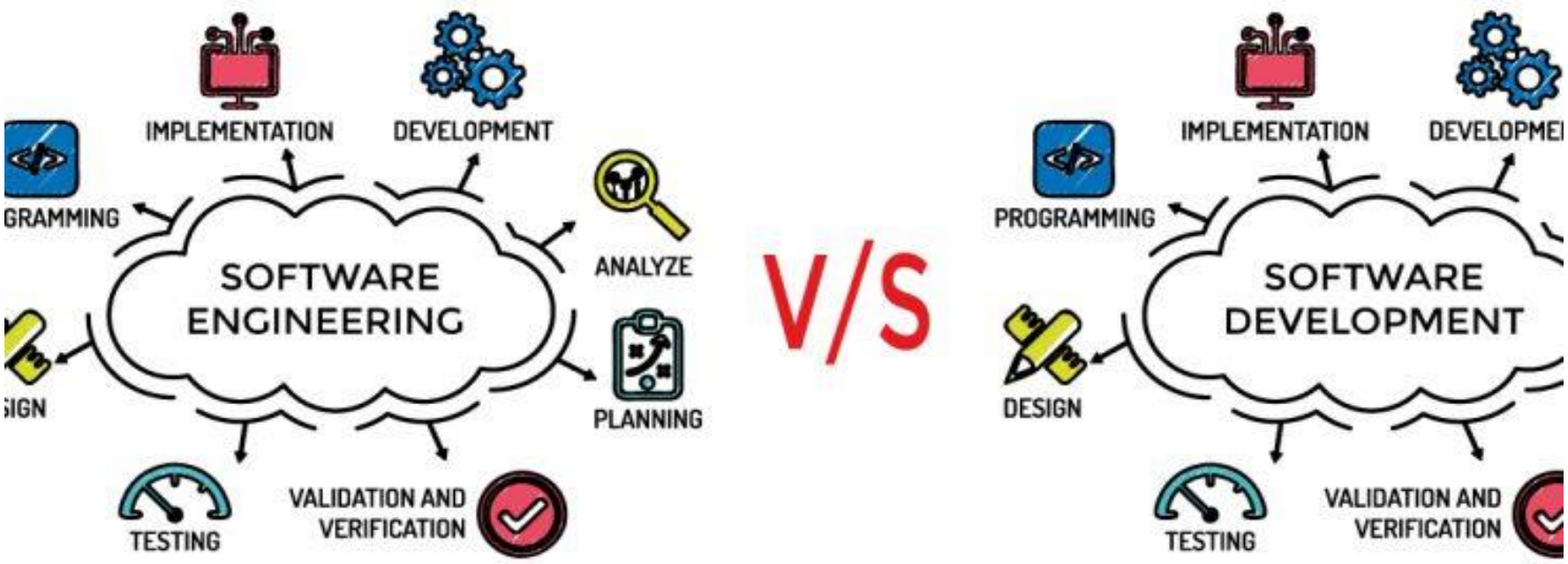
Lecture # 1

Lecture by Engr. Sidra

Objectives

- The objective of this lecture is to
  - Understand what software engineering is and why it is important;
  - Understand that the development of different types of software systems may require different software engineering techniques;
  - Understand some ethical and professional issues that are important for software engineers;

Why we need to study SOFTWARE ENGINEEIRNG?



SOFTWARE + ENGINEERING = SOFTWARE ENGINEERING

What is Software?

- Computer programs and associated documentation.
- Software encompasses:
  - **Instructions** (computer programs) that when executed provide desired features, function, and performance;
  - **Data structures** that enable the programs to adequately store and manipulate information and
  - **Documentation** that describes the operation and use of the programs
- The product that software professionals build and then support over the long term.

Software costs

- Software costs often dominate computer system costs. The costs of software on a PC are often greater than the hardware cost.
- Software costs more to maintain than it does to develop. For systems with a long life, maintenance costs may be several times development costs.
- Software engineering is concerned with cost-effective software development

Software Products

- Generic products
  - Particular systems that are marketed and sold to any customer who wishes to buy them.
  - The specification of what the software should do is owned by the software developer and decisions on software change are made by the developer.
  - Examples – PC software such as word processors, graphics programs, project management tools; software for specific markets such as appointments systems for dentists.

Software Products

- Customized products
  - Software that is commissioned by a specific customer to meet their own needs.
  - The specification of what the software should do is owned by the customer for the software and they make decisions on software changes that are required
  - Examples – control systems for electronic devices, systems written to support particular business process, air traffic control software

Generic Software vs Custom Software

Generic Software

- ✓Less expensive
- ✓More reliable
- ✓Immediate installation
- ✓More user friendly
- ✓User requirements are not completely satisfied
- ✓Cannot accommodate future changes

Custom Software

- ✓More expensive
- ✓Less reliable
- ✓Delay due to high development time
- ✓Less user friendly
- ✓Completely satisfied
- ✓Can easily accommodate future changes

Essential attributes of good software

Product Characteristics	Description
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
Dependability and Security	Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilization, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use

Software Engineering Definition

- The seminal definition:
  - [Software engineering is] the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.
- The IEEE definition:
  - The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software and the study of approaches as in.

Importance of Software Engineering

- More and more, individuals and society rely on advanced software systems. We need to be able to produce reliable and trustworthy systems economically and quickly.
- It is usually cheaper, in the long run, to use software engineering methods and techniques for software systems rather than just write the programs as if it was a personal programming project. For most types of system, the majority of costs are the costs of changing the software after it has gone into use

Importance of Software Engineering

- Reduce complexity
- To minimize software cost
- To decrease time
- Handling big projects
- Reliable software
- Effectiveness