Outline – Week 1

- What is Software?
- What is Software engineering?
- Why Software Engineering?
- Software development methods
What is Software?

- 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.
What is Good Software?

- Software has number of attributes which decide whether it is a good or bad.
- The definition of a good software changes with the person who evaluates it.
- The software is required by the customer, used by the end users of an organization and developed by software engineer.
- Each one will evaluate the different attributes differently in order to decide whether the software is good.
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**
  - Software must be trustworthy

- **Efficiency**
  - Software should not make wasteful use of system resources

- **Usability**
  - Software must be usable by the users for which it was designed
Software products

- **Generic products**
  - Stand-alone systems that are marketed and sold to any customer who wishes to buy them.
  - Examples – PC software such as editing, graphics programs, project management tools; CAD software; software for specific markets such as appointments systems for dentists.

- **Customized products**
  - Software that is commissioned by a specific customer to meet their own needs.
  - Examples – embedded control systems, air traffic control software, traffic monitoring systems.
Types of Software

- **System Software** - A collection of programs written to service other programs at system level. For example, compiler, operating systems.

- **Real-time Software** - Programs that monitor/analyze/control real world events as they occur.

- **Business Software** - Programs that access, analyze and process business information.

- **Engineering and Scientific Software** - Software using “number crunching” algorithms for different science and applications. System simulation, computer-aided design.
Types of Software

- Embedded Software:
  - Embedded software resides in read-only memory and is used to control products and systems for the consumer and industrial markets. It has very limited and esoteric functions and control capability.

- Artificial Intelligence (AI) Software:
  - Programs make use of AI techniques and methods to solve complex problems. Active areas are expert systems, pattern recognition, games
Types of Software

- Internet Software:
  - Programs that support internet accesses and applications. For example, search engine, browser, e-commerce software, authoring tools.

- Software Tools and CASE environment:
  - Tools and programs that help the construction of application software and systems. For example, test tools, version control tools.
Why Software is Important?

- The economies of ALL developed nations are dependent on software.
- More and more systems are software controlled (transportation, medical, telecommunications, military, industrial, entertainment,)
- Software engineering is concerned with theories, methods and tools for professional software development.
- Expenditure on software represents a significant fraction of GNP in all developed countries.
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.
Poor Software is a Concern: Space Shuttle Example

- Cost: Billions, millions of dollars more than planned
- Time: 3 years late
- Quality: First launch of Columbia was cancelled because of a synchronization problem with the Shuttle's 5 onboard computers.
- Substantial errors still exist.
  - Astronauts are supplied with a book of known software problems "Program Notes and Waivers".
The goal is to produce high quality software to satisfy a set of functional and nonfunctional requirements. How do we do that?

First, and foremost, by acknowledging that it is a problem solving activity.

It has to rely on well known techniques that are used all over the world for solving problems.
SE: A Problem Solving Activity

- **Analysis:**
  - Understand the nature of the problem

- **Synthesis:**
  - Put the pieces together into a large structure

- **Techniques (methods):**
  - Formal procedures for producing results using some well-defined notation

- **Methodologies:**
  - Collection of techniques applied across software development

- **Tools:**
  - Instrument or automated systems to accomplish a technique
Software Engineering: Definition

- Software Engineering is a collection of techniques, methodologies and tools that help with the production of a high quality software system with a given budget before a given deadline while change occurs.
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.
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is software?</td>
<td>Computer programs, data structures and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.</td>
</tr>
<tr>
<td>What are the attributes of good software?</td>
<td>Good software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable.</td>
</tr>
<tr>
<td>What is software engineering?</td>
<td>Software engineering is an engineering discipline that is concerned with all aspects of software production.</td>
</tr>
<tr>
<td>What is the <strong>difference</strong> between software engineering and computer science?</td>
<td>Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.</td>
</tr>
<tr>
<td>What is the <strong>difference</strong> between software engineering and system engineering?</td>
<td>System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is part of this more general process.</td>
</tr>
</tbody>
</table>
Why Software Engineering?

- Objectives:
  - Identify new problems and solutions in software production.
  - Study new systematic methods, principles, approaches for system analysis,
  - Design, implementation, testing and maintenance.
  - Provide new ways to control, manage, and monitor software process.
  - Build new software tools and environment to support software engineering.
Why Software Engineering?

- Major Goals:
  - To increase software productivity and quality.
  - To effectively control software schedule and planning.
  - To reduce the cost of software development.
  - To meet the customers’ needs and requirements.
  - To enhance the conduction of software engineering process.
  - To improve the current software engineering practice.
  - To support the engineers’ activities in a systematic and efficient manner.
Scientist Vs. Engineer

- **Computer Scientist**
  - Proves theorems about algorithms, designs languages, defines knowledge representation schemes
  - Has infinite time…

- **Engineer**
  - Develops a solution for a specific problem for a client
  - Uses computers & languages, tools, techniques and methods
  - Has finite (usually enough) time…

- **Software Engineer**
  - Works in multiple application domains
  - Has limited time to produce a software application
Programming Vs. Software Engineering

- **Programming**
  - The process of translating a problem from its physical environment into a language that a computer can understand and obey. (Webster’s New World Dictionary of Computer Terms)
  - The art of debugging a blank sheet of paper.
  - A pastime similar to banging one's head against a wall, but with fewer opportunities for rewards. (2 and 3 from The New Hacker’s Dictionary)

- **Software Engineering (according to Fritz Bauer)**
  - “The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.”
Software Engineering Vs. computer science?

Computer Science

- theory
- fundamentals

Software Engineering

- the practicalities of developing
- delivering useful software

*Computer science theories* are currently insufficient to act as a complete underpinning for software engineering, BUT it is a foundation for practical aspects of software engineering.
Software engineering Vs. System Engineering

- Software engineering is part of System engineering
- System engineering is 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
SE approach has two components, namely systems engineering approach and development engineering approach. The software and its quality depends upon the system in which it is installed.

The system here has a broad meanings. The understanding of the system can be achieved by the System study and Analysis.

- The System study and Analysis is carried out through SEM (Systems Engineering and Methodology). The SEM steps are as under:
  - Define the Objective of the system
  - Define the boundaries of the system
Components of Software Engineering

- Factories the system into different components
- Understand the relationship between various components
- Define relationship in terms of inputs, outputs and processes
- Understand the role of hardware and software
- Identify the key operational and functional requirements
- Model the system for analysis and development
- Discuss the system with the customer
Components of Software Engineering

- Development Engineering methodology has of translating the system requirements as software system goal, and proceeds to achieve it through a series of steps. The development engineering steps are:
  - Requirement definition and specification
  - Design solution to deliver the requirements
  - Determine the architecture for the delivery of solution
  - Customer development and planning
  - Software testing components
  - Integration of system components
  - Implementation
Components of Software Engineering

- Software development engineering is carried out in two ways
  - Structured System Analysis and Design (SSAD)
  - Object Oriented System Analysis and Design (OOSAD)

- Structured System Analysis and Design (SSAD)
  - The SSAD approach in which the system and its requirements are decomposed in a structured manner. Software development is carried out using sub-system structure, tested and integrated and implemented.
Object Oriented System Analysis and Design (OOSAD)

In contrast, the OOSAD development approach recommended the analysis of domain and builds objects of model independent of the system under consideration.

The object could represents a function, process or document evolved for the organization. Each object has attributes that describes the methods to perform and relationship to other objects.
Comparison between SSAD And OOSAD

- In SSAD the focus is on the functions and the data structure designed for those functions. Functions, data and processing methods are closely coupled. In OOSAD, however, objects and processing methods are decoupled from the data.

- In SSAD, skill lies in decomposing the system whereas in OOSAD skill lies in modeling the organization and its business in the objects.

- SSAD and OOSAD are dissimilar in focus but similar in that both propose a problem solving methodology and a set of techniques and tools to assist the S/W engineer analyze, model, design and develop the system.
The Role of Software Engineering

A bridge from customer needs to programming implementation

First law of software engineering
Software engineer is willing to learn the problem domain
(problem cannot be solved without understanding it first)
The Role of Software Engineering

Customer:
Requires a computer system to *achieve some business goals* by user interaction or interaction with the environment in a specified manner.

Software Engineer’s task:
To *understand how* the system-to-be needs to interact with the user or the environment so that customer’s requirement is met and *design* the software-to-be.

Programmer’s task:
To *implement* the software-to-be designed by the software engineer.

User

Environment

System-to-be

Software-to-be

May be the same person
Software System Quality Factors

- **Complexity:**
  - The system is so complex that no single programmer can understand it anymore
  - The introduction of one bug fix causes another bug

- **Change:**
  - The “Entropy” of a software system increases with each change: Each implemented change erodes the structure of the system which makes the next change even more expensive (“Second Law of Software Dynamics”).
  - As time goes on, the cost to implement a change will be too high, and the system will then be unable to support its intended task. This is true of all systems, independent of their application domain or technological base.
Software is Complex

- Software system is complex but not complicated
  - Complex = composed of many simple parts related to one another
  - Complicated = not well understood, or explained
Complexity Example: Scheduling Fence Construction Tasks

Setting posts [3 time units]
Cutting wood [2 time units]
Nailing [2 time units for unpainted; 3 time units otherwise]
Painting [5 time units for uncut wood; 4 time units otherwise]

Setting posts < Nailing, Painting
Cutting < Nailing

...shortest possible completion time = ?

[⇒ “simple” problem, but hard to solve without a pen and paper]
Suppose today is Tuesday, January 14, 2014

What day will be on January 23, 2017?

[To answer, we need to bring the day names and the day numbers into coordination, and for that we may need again a pen and paper]
Why are software systems so complex?

- The problem domain is difficult
- The development process is very difficult to manage
- Software offers extreme flexibility
- Software is a discrete system
  - Continuous systems have no hidden surprises (Parnas)
  - Discrete systems have!
Method = work strategy

- The Feynman Problem-Solving Algorithm:
  (i) Write down the problem (ii) think very hard, and (iii) write down the answer.

Waterfall

- Unidirectional, finish this step before moving to the next

Iterative + Incremental

- Develop increment of functionality, repeat in a feedback loop

Agile

- User feedback essential; feedback loops on several levels of granularity