Fundamentals of Software Engineering
Unit – I
Topic-I

Introduction

- Software is a product
  - ➢ Delivers computing potential
  - ➢ Produces, manages, acquires, modifies, displays, or transmits information
- Software is a vehicle for delivering a product
  - ➢ Supports or directly provides system functionality
  - ➢ Controls other programs (e.g., an operating system)
  - ➢ Effects communications (e.g., networking software)
  - ➢ Helps build other software (e.g., software tools)
- Software is a set of items or objects that form a “configuration” that includes
  - Programs
  - Documents
  - Data

Software Characteristics

- Software Is Engineered
- Software Doesn’t Wear Out
- Software Is Complex

Software Applications

- System software
- Application software
- Engineering/scientific software
- Embedded software
- Product-line software
- Web applications
- AI software

Software- Current Categories

- Ubiquitous computing: Wireless Networks
- Net-sourcing: Web as a computing engine
- Open source: “free” source code open to the computing community
- Data mining
- Grid computing
- Cognitive machines
- Software for nanotechnologies

Need for Software engineering

- The economies of ALL developed nations are dependent on software
- More and more systems are software controlled
- Software engineering is concerned with theories, methods and tools for professional software development
- Software engineering expenditure represents a significant fraction of GNP in all developed countries

Faqs About Software Engineering

- Define Software.
- What is software engineering?
- What is the difference between software engineering and computer science?
- What is the difference between software engineering and system engineering?
- What is a software process?
- What is a software process model?
- What are the costs of software engineering?
- What are software engineering methods?
- What is CASE (Computer-Aided Software Engineering)
- What are the attributes of good software?
- What are the key challenges facing software engineering?

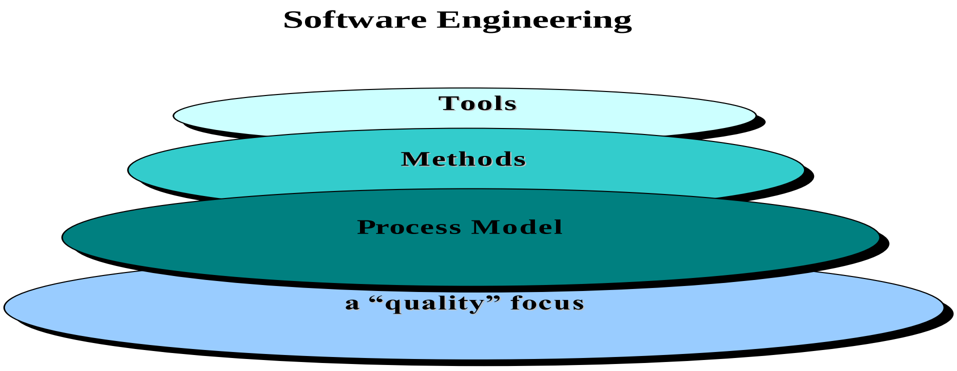
Definition of Software

- Computer **programs** + 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
  - Bespoke (custom) - developed for a single customer according to their specification

What Is Software Engineering?

- Software engineering is an engineering discipline which is concerned with all aspects of software production
- According to IEEE, the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of **<u>software</u>**"

SE as LayeredTechnology



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.

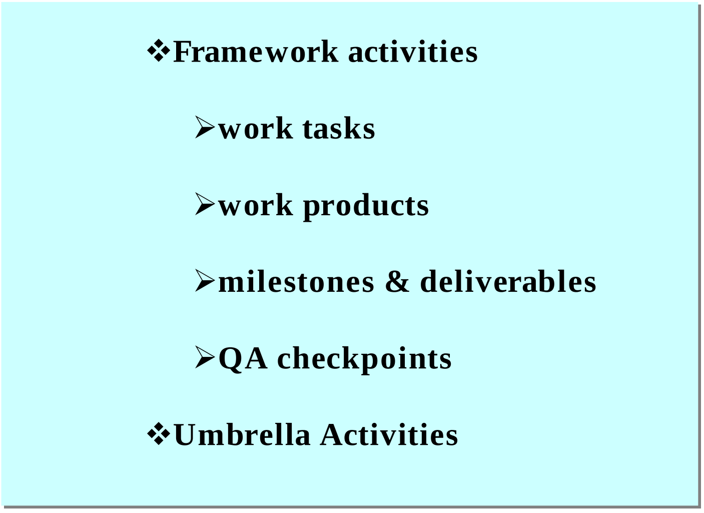
Difference between Software Engineering and System Engineering

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

Software process

- A set of activities whose goal is the development or evolution of software
- Generic activities in all software processes are:
  - ➢ Specification - what the system should do and its development constraints
  - ➢ Development - production of the software system
  - ➢ Validation - checking that the software is what the customer wants
  - ➢ Evolution - changing the software in response to changing demands

A Process Framework



- ❖**Framework activities**
  - ➢**work tasks**
  - ➢**work products**
  - ➢**milestones & deliverables**
  - ➢**QA checkpoints**
- ❖**Umbrella Activities**

Framework Activities

- Communication
- Planning
- Modeling
  - ➢Analysis of requirements
  - ➢Design
- Construction
  - ➢Code generation
  - ➢Testing
- Deployment

Umbrella Activities

- Software project management
- Formal technical reviews
- Software quality assurance
- Software configuration management
- Work product preparation and production
- Reusability management
- Measurement
- Risk management

Software Process Model

- A simplified representation of a software process, presented from a specific perspective.
- Examples of process perspectives are:
  - ➢ Workflow perspective - sequence of activities
  - ➢ Data-flow perspective - information flow
  - ➢ Role/action perspective - who does what

Costs of 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.

Software Engineering Methods

- Structured approaches to software development which include system models, notations, rules, design advice and process guidance.
- Model descriptions
  - Descriptions of graphical models which should be produced
- Rules
  - Constraints applied to system models
- Recommendations
  - Advice on good design practice
- Process guidance
  - What activities to follow

CASE (Computer-Aided Software Engineering)

- Software systems which are intended to provide automated support for software process activities.
- CASE systems are often used for method support
- Upper-CASE
  - Tools to support the early process activities of requirements and design
- Lower-CASE
  - Tools to support later activities such as programming, debugging and testing

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

Key challenges of Software Engineering

- Coping with legacy systems, coping with increasing diversity and coping with demands for reduced delivery times
- Legacy systems
  - Old, valuable systems must be maintained and updated
- Heterogeneity
  - Systems are distributed and include a mix of hardware and software
- Delivery
  - Increasing pressure for faster delivery of software

Professional and Ethical Responsibility

- Software engineering involves wider responsibilities than simply the application of technical skills
- Software engineers must behave in an honest and ethically responsible way if they are to be respected as professionals
- Ethical behaviour is more than simply upholding the law.

Issues of Professional Responsibility

- *Confidentiality*
  - Engineers should normally respect the confidentiality of their employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.
- *Competence*
  - Engineers should not misrepresent their level of competence. They should not knowingly accept work which is out with their competence.