UNIT-1 INTRODUCTION TO SOFTWARE AND SOFTWARE ENGINEERING

Part-1

Software Engineering: Introduction

- Software has become critical to advancement in almost all areas of human endeavour.
- The art of programming only is no longer sufficient to construct large programs.
- There're serious problems in the cost, timeliness, maintenance, and quality of many software products.
- Software engineering has the objective of solving these problems by
  - producing good quality,
  - maintainable software,
  - on time,
  - Within budget

Software Engineering: Introduction

- The term is made of two words, software and engineering.
  - Software is a general term for the various kinds of programs (specific set of ordered operations) used to operate computers and related devices.
  - **Engineering** is the analysis, design, construction, verification, and management of technical (or social) entities using scientific methodologies.

Note: Difference: Program (Single developer, Lack proper interface & documentation)and Software (Team developer, Well defined interface and documentation)

Software Components

- Software comprises of the following components:

  1. Programs
  2. Documents
  3. Operating Procedures

- Programs are the source codes.
- Documents comprise of manuals created at the end of every phase. The following documents are developed at the end of every phase:

Phase	Document
1. Analysis	SRS, DFD, Context Diagram, etc.
2. Design	Flowchart, ER Diagram, etc.
3. Coding	Source Code Listing, Reference Listing.
4. Testing	Test data, test cases and results.

Software Components

- Procedures can be given in form of manuals.

The following procedural documents are developed: **Procedural Document Type Description**

1. User Manual System Overview
   Beginner'sManual/Guide
   Reference Manual
2. Operational Manual Installation Guide
   Troubleshooting Guide

Software Engineering: Definition

- Fritz Bauer, a German computer scientist, defines software engineering as:

  Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and work efficiently on real machines.

- IEEE defines software engineering as:

  (1) The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software; that is, the application of engineering to software.

  (2) The study of approaches as in the above statement.

Generic View of Software Engineering

- The work associated with software engineering can be categorized into three generic phases, regardless of application area, project size, or complexity.
- **The *definition phase*** *focuses on what. That is, during definition, the software engineer attempts to identify what information is to be processed, what function and performance are desired, what system behavior can be expected, what interfaces are to be established, what design constraints exist, and what validation criteria are required to define a successful system.*
- **The *development phase*** *focuses on how. That is, during development a software engineer attempts to define how data are to be structured, how function is to be implemented within a software architecture, how procedural details are to be implemented, how interfaces are to be characterized, how the design will be translated into a programming language (or nonprocedural language), and how testing will be performed.*
- **The *support phase*** *focuses on change associated with error correction, adaptations required as the software's environment evolves, and changes due to enhancements brought about by changing customer requirements.*

The Evolving Role of Software

- The industry originated with the entrepreneurial computer software and services companies of the 1950s and 1960s, grew dramatically through the 1970s and 1980s to become a market force rivaling that of the computer hardware companies.
- And by the 1990s had become the supplier of technical know-how that transformed the way people worked, played and communicated every day of their lives.

The Evolving Role of Software

- Today, software takes on a dual role. It is a product and, at the same time, the vehicle for delivering a product.
  - As a product, it delivers the computing potential embodied by computer hardware or, more broadly, a network of computers that are accessible by local hardware.
  - As the vehicle used to deliver the product, software acts as the basis for the control of the computer (operating systems), the communication of information (networks), and the creation and control of other programs (software tools and environments).

Software Myths

- Software myths are misleading attitudes that have caused serious problems for managers and technical people alike. Software myths propagate misinformation and confusion. There are three kinds of software myths:
  1. Management myths
  2. Customer myths
  3. Practitioner's myths

Software Myths

- Management myths: Managers with software responsibility are often under pressure to maintain budgets, keep schedules from slipping, and improve quality. Following are the management myths:
  - Myth: We already have a book that's full of standards and procedures for building software, won't that provide my people with everything they need to know?
  - Reality: The book of standards may very well exist, but isn’t used. Most software practitioners aren’t aware of its existence. Also, it doesn’t reflect modern software engineering practices and is also complete.

Software Myths

- Myth: My people have state-of-the-art software development tools, after all, we buy them the newest computers.
- Reality: It takes much more than the latest model mainframe, workstation, or PC to do high-quality software development. Computer-aided software engineering (CASE) tools are more important than hardware for achieving good quality and productivity.

- Myth: If we get behind schedule, we can add more programmers and catch up (sometimes called the Mongolian horde concept).
- Reality: Software development is not a mechanistic process like manufacturing. As new people are added,

Software Myths

- Myth: If I decide to outsource the software project to a third party, I can just relax and let that firm build it.
- Reality: If an organization does not understand how to manage and control software projects internally, it will invariably struggle when it outsources software projects.

- **2) Customer myths:** Customer myths lead to false expectations (by the customer) and ultimately, dissatisfaction with the developer. Following are the customer myths:
- Myth: A general statement of objectives is sufficient to begin writing programs-we can fill in the details later.
- Reality: A poor up-front definition is the major cause of

Software Myths

- Myth: Project requirements continually change, but change can be easily accommodated because software is flexible.
- Reality: It is true that software requirements change, but the impact of change varies with the time at which it is introduced. When changes are requested during software design, the cost impact grows rapidly. Resources have been committed and a design framework has been established. Change can cause heavy additional costs. Change, when requested after software is in production, can be much more expensive than the same change requested earlier.

Software Myths

- 3) Practitioner's myths: Practitioners have following myths:
- Myth: Once we write the program and get it to work, our job is done.
- Reality: Industry data indicate that between 60 and 80 percent of all effort expended on software will be expended after it is delivered to the customer for the first time.

- Myth: Until I get the program "running" I have no way of assessing its quality.
- Reality: One of the most effective software quality assurance mechanisms can be applied from the

Software Myths

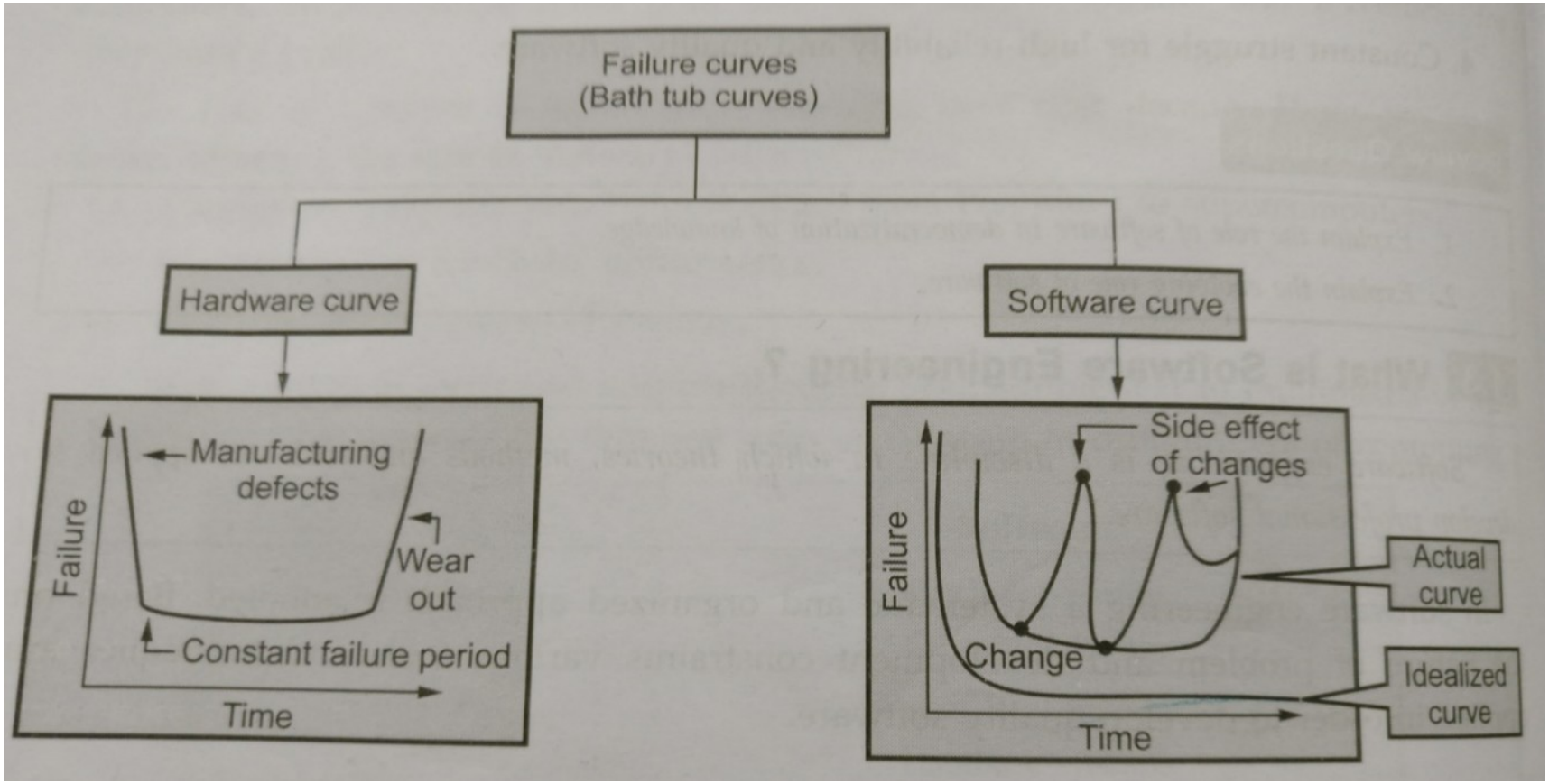
- Myth: The only deliverable work product for a successful project is the working program.
- Reality: A working program is only one part of a software configuration that includes many elements. Documentation provides a foundation for successful engineering and, more important, guidance for software support.

- Myth: Software engineering will make us create voluminous and unnecessary documentation and will invariably slow us down.
- Reality: Software engineering is not about creating documents. It is about creating quality. Better quality leads to reduced rework. And reduced rework results in faster delivery times.

Characteristics of Software

1. *Software is developed or engineered, it is not manufactured in the classical sense.*
   - Although some similarities exist between software development and hardware manufacture, the two activities are fundamentally different.
   - In both activities, high quality is achieved through good design, but the manufacturing phase for hardware can introduce quality problems that are nonexistent (or easily corrected) for software.

Characteristics of Software



Characteristics of Software

2. Software doesn't "wear out."

- hardware components suffer from the cumulative affects of just, vibration, abuse, temperature extremes, and many other environmental changes. The hardware begins to wear out.
- Software is not susceptible to the environmental changes that cause the hardware to wear out.

3. Although the industry is moving toward component-based assembly, most software continues to be custom built.

While developing any hardware product, firstly the circuit design with desired functioning properties is created. Then required hardware components such as ICs, capacitors and registers are assembled according to the design, but this is not done while developing software product. Most of the software is custom built.

Software Engineering as a Layered Technology

Figure: Layered Technology of Software Engineering

Software Engineering as a Layered Technology

- Software engineering is a layered technology encompasses of a process, management & technical methods, and tools.
- Any engineering approach (including software engineering) must rest on an organizational commitment to quality. The bedrock that supports software engineering is a quality focus.

Software Engineering as a Layered Technology

- Process:
  - The foundation for software engineering is the process layer.
  - The Software engineering process is the glue that holds the technology layers together and enables rational and timely development of computer software.
  - Process defines a framework that must be established for effective delivery of software engineering technology and forms the basis for management & control of software projects.

Software Engineering as a Layered Technology

- Methods:
  - Software engineering methods provide the technical how-to's for building software.
  - Methods encompass a broad array of tasks that include requirements analysis, design, program construction, testing, and support.

Software Engineering as a Layered Technology

- Tools:
  - Software engineering tools provide automated or semi-automated support for the process and the methods.
  - When tools are integrated so that information created by one tool can be used by another, a system for the support of software development, called computer-aided software engineering (CASE), is established.
  - **CASE combines software, hardware, and a software engineering database** (a repository containing important information about analysis, design, program construction, and testing)

Software Applications

- The role of software Engineer is to plan and coordinate software system for a specified purpose.

- System software: System software is a collection of programs written to service other programs.
  - Some system software (e.g., compilers, editors, and file management utilities) process complex, but determinate, information structures.
  - Other systems applications (e.g., operating system components, drivers, telecommunications processors) process largely indeterminate data.

Software Applications

- Real-time software: Software that monitors/analyzes/controls real-world events as they occur is called real time.
  - Elements of real-time software include
    - a data gathering component that collects and formats information from an external environment,
    - an analysis component that transforms information as required by the application,
    - a control/output component that responds to the external environment,
    - and a monitoring component that coordinates all other components so that real-time response (typically ranging from 1 millisecond to 1 second) can be maintained.

Software Applications

- Business software: Business information processing is the largest single software application area.
  - Discrete "systems" (e.g., payroll, accounts receivable/payable, inventory) have evolved into management information system (MIS) software that accesses one or more large databases containing business information.
- Engineering and scientific software: Engineering and scientific software have been characterized by "number crunching" algorithms.
  - Applications range from astronomy to volcanology, from automotive stress analysis to space shuttle orbital dynamics, and from molecular biology to automated manufacturing.

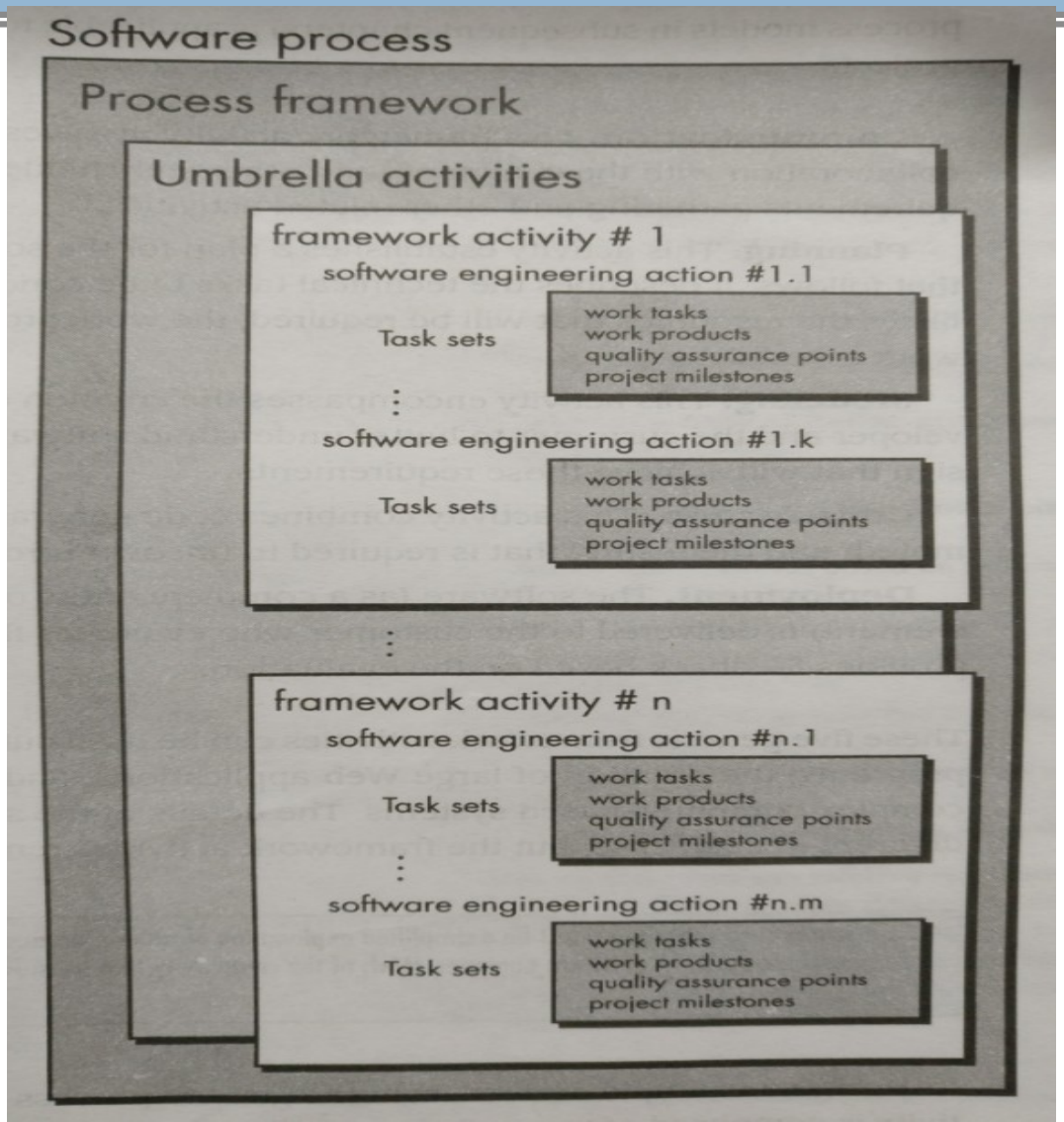
Software Applications

- Embedded software: Intelligent products have become commonplace in nearly every consumer and industrial market. Embedded software resides in read-only memory and is used to control products and systems for the consumer and industrial markets.
  - Embedded software can perform very limited and esoteric functions (e.g., keypad control for a microwave oven) or provide significant function and control capability (e.g., digital functions in an automobile such as fuel control, dashboard displays, and braking systems).
- Personal computer software: The personal computer software market has burgeoned over the past two decades.
  - Word processing, spreadsheets, computer graphics, multimedia, entertainment, database management, personal and business financial applications, external network, and database access are only a few of hundreds of applications.

Process Framework

- Software process can be defined as the structured set of activities that are required to develop the software system.
- The process framework is required for representing the common process activities.
- As shown in figure software process is characterized by process framework activities, umbrella activities and task sets.

Process Framework



Process Framework

- Process Framework activities:
  - Communication
  - Planning
  - Modeling
  - Construction
  - Deployment

Task Set:

A Task set defines the actual work to be done to accomplish the objectives of a software engineering action.

Process Framework

- Umbrella activities:
  - Software project tracking and control
  - Risk management
  - Software Quality assurance
  - Formal technical reviews
  - Measurement
  - Software configuration management
  - Reusability management
  - Work product preparation and production

Product and Process

Sr. No.	Software process	Software product
1.	Processes are developed by individual user and it is used for personal use.	Software product is developed by multiple users and it is used by large number of people or customers.
2.	Process may be small in size and possessing limited functionality. It defines framework for effective product generation.	Software product consists of multiple program codes, related documents such as SRS, designing documents user manuals, test cases and so on.
3.	Generally only one person uses the process, hence there is a lack of user interface.	Good graphical user interface is most required by any software product.
4.	Process is generally developed by process engineers.	Software product is developed by **software engineers** who are large in number and work in a team. Therefore systematic approach of developing software product must be applied.
5.	The output of process is product.	Product development occurs by following process.
6.	**For example :** Program developed for parsing the input.	**For example :** A word processing software.