# Introduction to Software Engineering

Prof. Vidula. V. Meshram
Vidula.meshram@viit.ac.in
**Department of Computer Engineering**


**BRACT'S, Vishwakarma Institute of Information Technology, Pune-48**

(An Autonomous Institute affiliated to Savitribai Phule Pune University)
(NBA and NAAC accredited, ISO 9001:2015 certified)

# Objective/s of this session

1. To understand nature of software
2. To learn software engineering principles and software processes.
3. To discuss myths about software

## Learning Outcome/Course Outcome

1. To understand software engineering principles and software processes
2. To know nature and myths of software

# Content

- Nature of Software
- Software Engineering Principles
- Software Process
- Software Myths

# What is a Software ?

 Software is:

(1) Instructions (computer programs) that when executed provide desired features, function, and performance

(2) Data structures that enable the programs to adequately manipulate information

(3) Documentation that describes the operation and use of the programs.

# What is software?

- Computer programs and associated documentation such as requirements, design models and user manuals.
- 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 e.g. PC software such as Excel or Word.
  - **Custom** - developed for a single customer according to their specification.
- New software can be created by developing new programs, configuring generic software systems or reusing existing software.

# Software's Dual Role

- **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)

# Characteristics of Software

- Software is developed or engineered, it is not manufactured in the classical sense.
- Software doesn't "wear out.“
- Although the industry is moving toward component-based construction, most software continues to be custom-built..

# Wear Vs. Deterioration

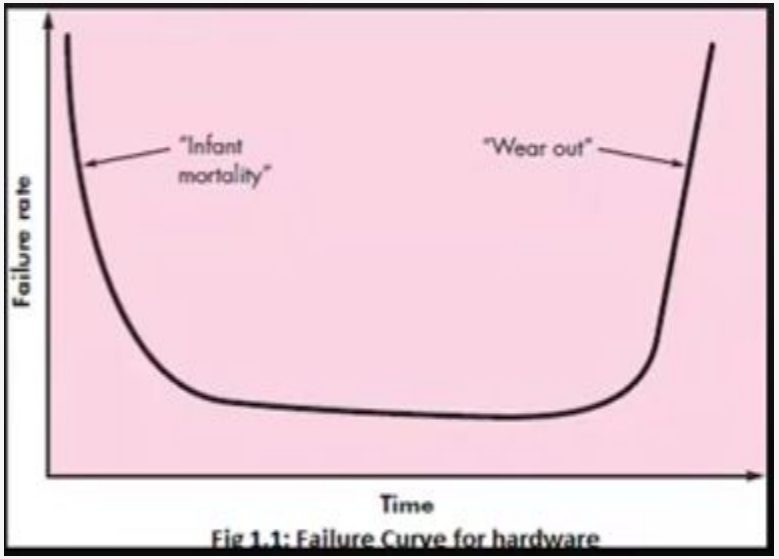


Fig 1.1: Failure Curve for hardware

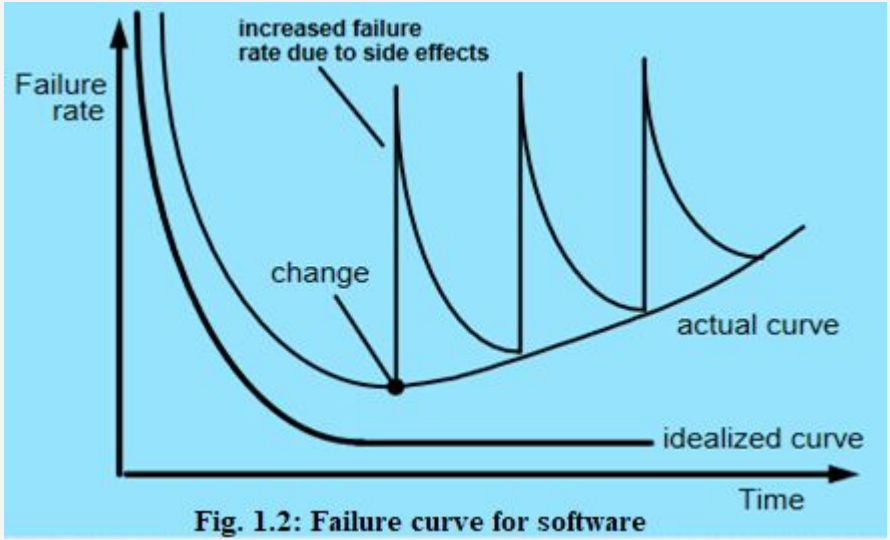


Fig. 1.2: Failure curve for software

# Software Application Domains

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

# Software—New Categories

- **Ubiquitous computing**—wireless networks
- **Netsourcing**—the Web as a computing engine
- **Open source**—”free” source code open to the computing community (a blessing, but also a potential curse!)
- **Data mining**
- **Grid computing**
- **Cognitive machines**
- **Software for nanotechnologies**

# Legacy Software

- Software must be **adapted** to meet the needs of new computing environments or technology.
- Software must be **enhanced** to implement new business requirements.
- Software must be **extended to make it interoperable** with other more modern systems or databases.
- Software must be **re-architected** to make it viable within a network environment**.**

# What is software engineering?

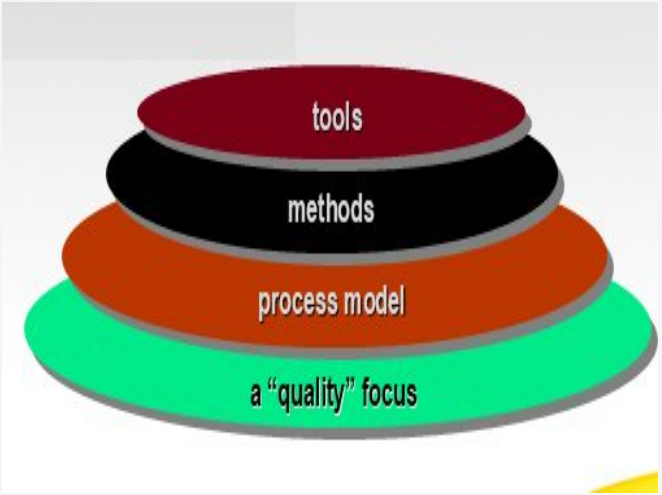
- Software engineering is an engineering discipline that is concerned with **all aspects of software production**.
- Software engineers should adopt a systematic and organised approach to their work and use appropriate tools and techniques depending on the problem to be solved, the development constraints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

❑ The IEEE definition:

- *Software Engineering: (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 (1).*

# Software Engineering- A Layered Technology

- Software Engineering(SE) approach has commitment to quality.
- Process Layer is foundation of Software Engineering. Process defines a framework that must be established for effective deliver of SE technology.
- SE methods provide technical how to's for building software.
- SE tools provide automated or semi automated support for the process and the methods. A system for the support of software development is called computer-aided software engineering (CASE)

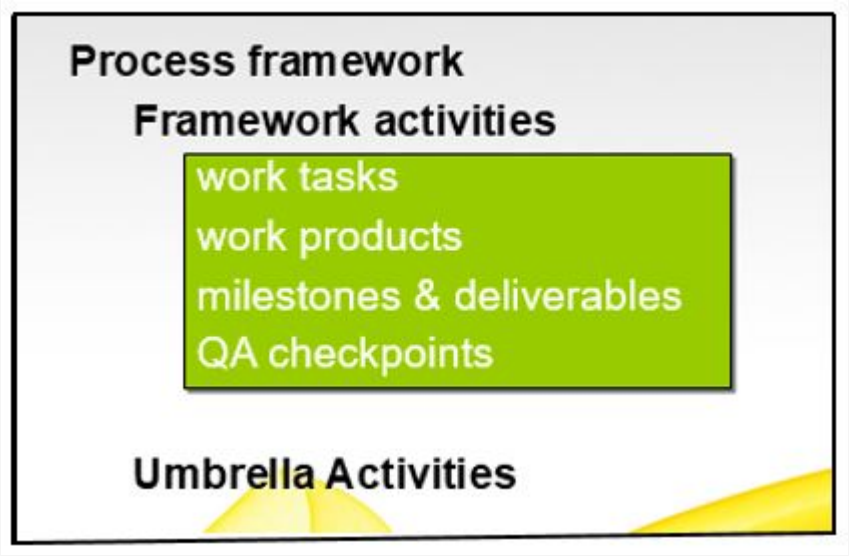



Software Engineering Layers

# Software Process

- A process is collection of ***activities***, ***actions*** and *tasks* that are performed when some work product is to be created.
- An ***activity*** tries to achieve a broad objective.
- An ***action*** encompasses a set of task that produce a major work product
- A ***task*** focuses on a small, but well-defined objective that producs some tangible outcome.

# Process Framework



# Process Framework Activities

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

# Umbrella Activities

- Software project tracking & control
- Risk management
- Software quality assurance
- Technical reviews
- Measurement
- Software configuration management
- Reusability management
- Work product preparation and production

# Attributes of good software

- The software should deliver the required functionality and performance to the user and should be maintainable, dependable and acceptable.
- Maintainability
  - Software must evolve to meet changing needs;
- Dependability
  - Software must be trustworthy;
- Efficiency
  - Software should not make wasteful use of system resources;
- Acceptability
  - Software must accepted by the users for which it was designed. This means it must be understandable, usable and compatible with other systems.

# Key challenges facing software engineering

- Heterogeneity, delivery and trust.
- Heterogeneity
  - Developing techniques for building software that can cope with heterogeneous platforms and execution environments;
- Delivery
  - Developing techniques that lead to faster delivery of software;
- Trust
  - Developing techniques that demonstrate that software can be trusted by its users.

# Software Myths

- It affects managers, customers (and other non-technical stakeholders) and practitioners
- These are believable because they often have elements of truth, but invariably lead to bad decisions,
- Therefore, it insists on reality as you navigate your way through software engineering

# 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 is it used ? Are software practitioners aware of its existence ? Does it reflect modern software development practice ? Is it complete ? In many cases, the answer to all of these questions is "no".

- **Myth :** My people do have state-of-the-art software development tools, after all, we buy them from the newest computers.
- **Reality :** It takes much more than the latest model mainframe, workstation, or PC to do high quality software development.

# Management Myths

- **Myth :** If we get behind schedule, we can add more programmers and catch up.
- **Reality :** Software development is not a mechanistic process like manufacturing. However, as new people are added, people who were working must spend time educating the newcomers, thereby reducing the amount of time spent on productive development effort.

# Customer Myths

- **Myth :** A general statement of objectives is sufficient to begin writing programs.
- **Reality :** Poor-up-front definition is the major cause of failed software efforts. A formal and detailed description of information domain, function, performance, interfaces, design constraints, and validation criteria is essential.

- **Myth :** Project requirements continuously change, but change can be easily accommodated because software is flexible.
- **Reality :** It is true that software requirements do change, but the impact of change varies with the time at which it is introduced.

# Practitioner Myths

- **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 a quality product. Better quality leads to reduced rework. And reduced rework results in faster delivery times.

- **Myth :** Once the program is written and running, my job is done.
- **Reality :** Someone once said that “the sooner you begin ‘writing code,’ the longer it’ll take you to get done.” 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.

# References


- Roger Pressman, “Software Engineering: A Practitioner’s Approach”, Mcgraw Hill
- Ian Sommerville, “ Software Engineering”, Addision and Wesley

# Thank You