# Introduction to Software Engineering

Instructor: Mehroze Khan

# What is 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.*

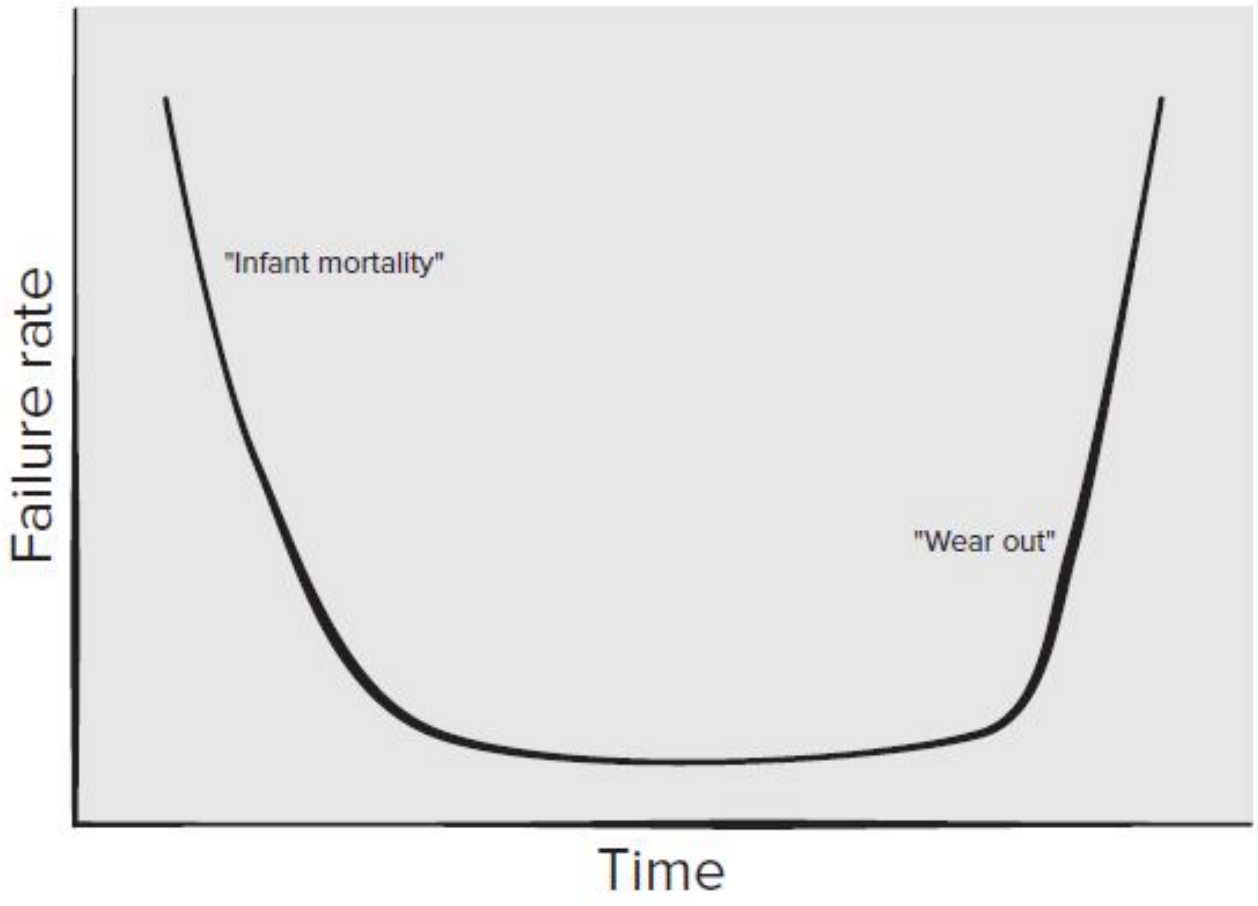
# What is Software?

***Software is developed or engineered, it is not manufactured in the classical sense.***

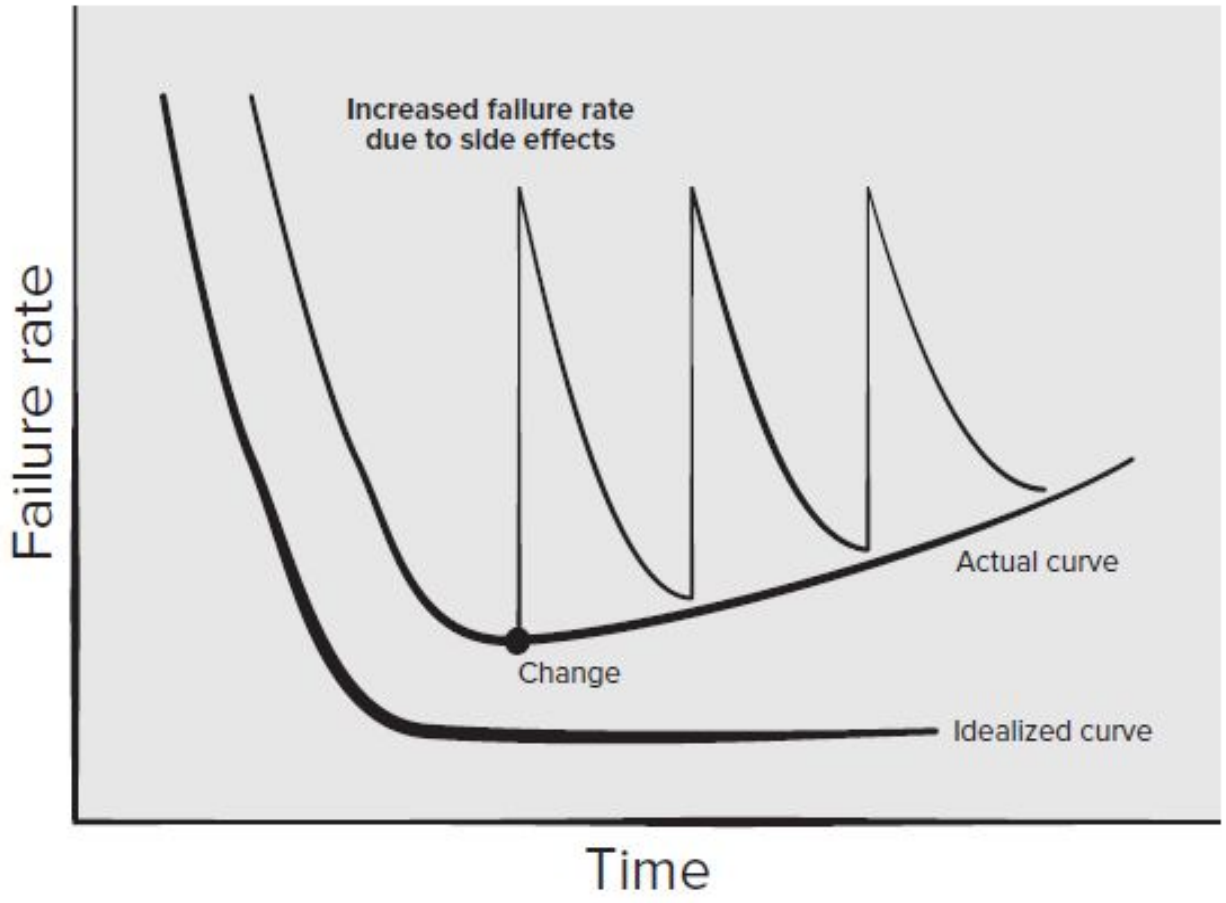
***Software doesn't "wear out". But it does deteriorate!***

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

# Failure Curve for Hardware



# Failure Curve for Software



# Software Application Domains

- System Software: A collection of programs written to service other programs. E.g., compilers, editors, file management utilities, networking software.
- **Application Software:** Stand-alone programs that solve a specific business need.
- **Engineering/Scientific Software:** A broad array of "number-crunching" or data science programs that range from astronomy to volcanology, from automotive stress analysis to orbital dynamics, from computer-aided design to consumer spending habits, and from genetic analysis to meteorology.
- **Embedded Software:** Resides within a product or system and is used to implement and control features and functions for the end user and for the system itself.

# Software Application Domains

- **Product-line Software:** Composed of reusable components and designed to provide specific capabilities for use by many different customers.
- **Web/Mobile Applications:** This network-centric software category spans a wide array of applications and encompasses browser-based apps, cloud computing, service-based computing, and software that resides on mobile devices.
- **Artificial Intelligence Software:** Makes use of heuristics to solve complex problems that are not amenable to regular computation or straightforward analysis. E.g., robotics, decision-making systems, pattern recognition (image and voice), machine learning, theorem proving, and game playing.

# Legacy Software

## *Why must it change?*

1

software must be adapted to meet the needs of new computing environments or technology.

2

software must be enhanced to implement new business requirements.

3

software must be extended to make it interoperable with other more modern systems or databases.

4

software must be re-architected to make it viable within a network environment.

# Software Engineering

**Some realities:**

- *a concerted effort should be made to understand the problem before a software solution is developed*
- *design becomes a pivotal activity*
- *software should exhibit high quality*
- *software should be maintainable*

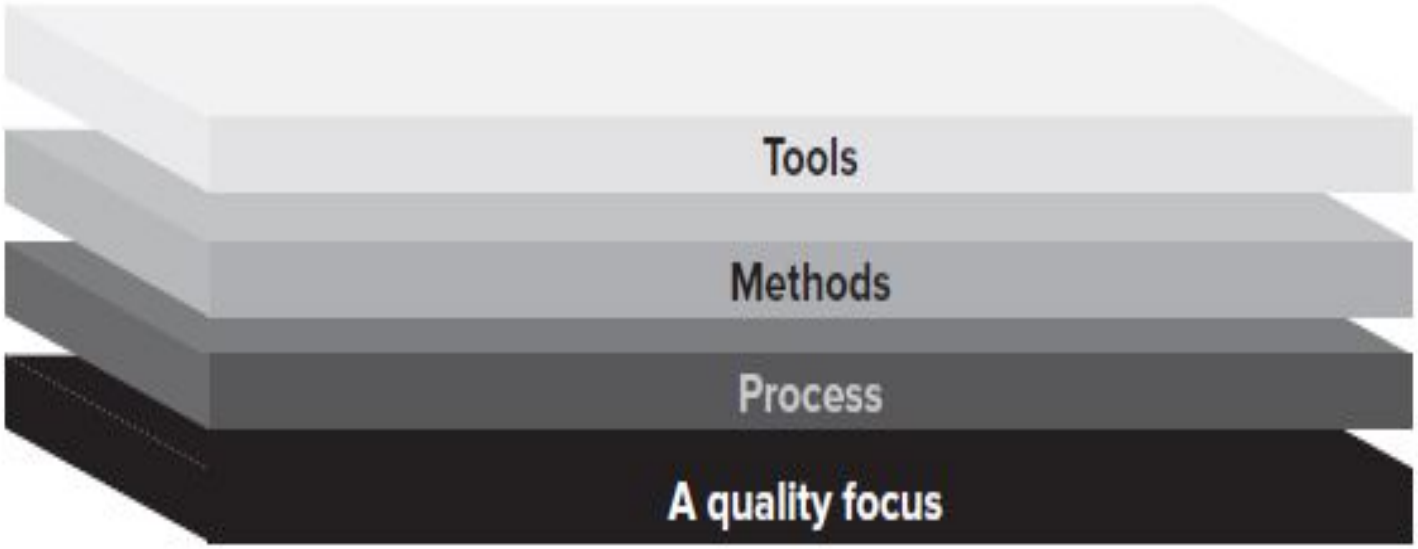
**The seminal definition:**

- *[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.*

# Software Engineering

- The IEEE definition:
  - *Software Engineering:*
    *The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software*

# A Layered Technology




*Software Engineering*

# Software Process

- A **process** is a collection of activities, actions, and tasks that are performed when some work product is to be created.
- An **activity** strives to achieve a broad objective (e.g. communication with stakeholders)
- An **action** encompasses a set of tasks that produce a major work product (e.g., an architectural design model)
- A **task** focuses on a small, but well-defined objective (e.g., conducting a unit test) that produces a tangible outcome.

# A Process Framework

**Process framework**

**Framework Activities**

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

# Adapting a Process Model

- the overall flow of activities, actions, and tasks and the interdependencies among them
- the degree to which actions and tasks are defined within each framework activity
- the degree to which work products are identified and required
- the manner which quality assurance activities are applied
- the manner in which project tracking and control activities are applied
- the overall degree of detail and rigor with which the process is described
- the degree to which the customer and other stakeholders are involved with the project
- the level of autonomy given to the software team
- the degree to which team organization and roles are prescribed

# The Essence of Practice

- Polya suggests:
  1. *Understand the problem* (communication and analysis).
  2. *Plan a solution* (modeling and software design).
  3. *Carry out the plan* (code generation).
  4. *Examine the result for accuracy* (testing and quality assurance).

# Understand the Problem

- *Who has a stake in the solution to the problem?* That is, who are the stakeholders?
- *What are the unknowns?* What data, functions, and features are required to properly solve the problem?
- *Can the problem be compartmentalized?* Is it possible to represent smaller problems that may be easier to understand?
- *Can the problem be represented graphically?* Can an analysis model be created?

# Plan the Solution

- *Have you seen similar problems before?* Are there patterns that are recognizable in a potential solution? Is there existing software that implements the data, functions, and features that are required?
- *Has a similar problem been solved?* If so, are elements of the solution reusable?
- *Can subproblems be defined?* If so, are solutions readily apparent for the subproblems?
- *Can you represent a solution in a manner that leads to effective implementation?* Can a design model be created?

# Carry Out the Plan

- *Does the solution conform to the plan?* Is source code traceable to the design model?
- *Is each component part of the solution provably correct?* Has the design and code been reviewed, or better, have correctness proofs been applied to algorithm?

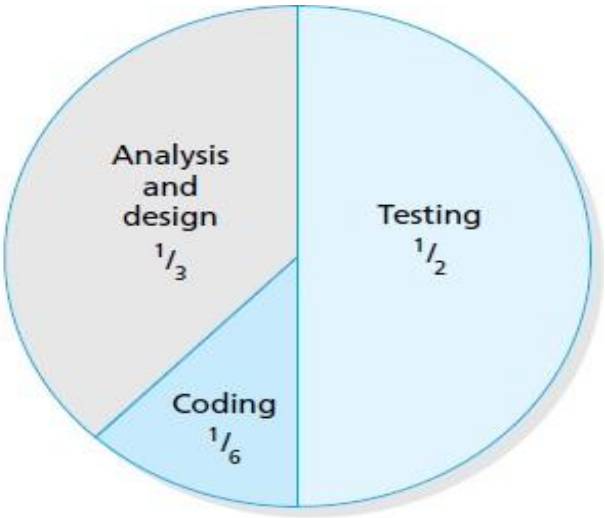
# Examine the Result

- *Is it possible to test each component part of the solution?* Has a reasonable testing strategy been implemented?
- *Does the solution produce results that conform to the data, functions, and features that are required?* Has the software been validated against all stakeholder requirements?

# Hooker's General Principles for SE Practice

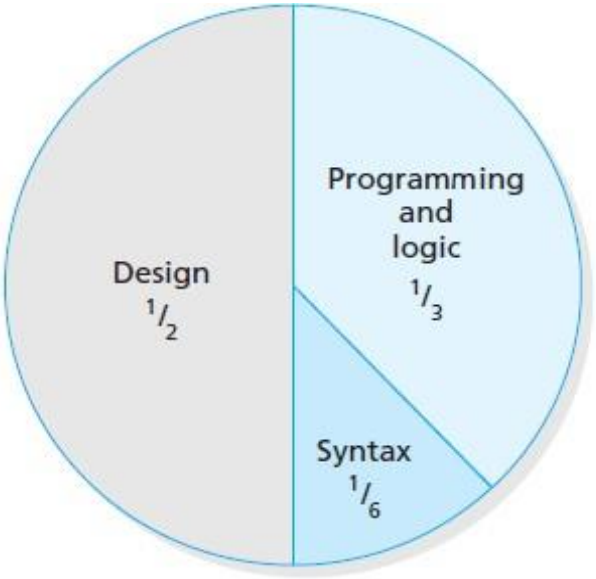
- 1: *The Reason It All Exists*
- 2: *KISS (Keep It Simple, Stupid!)*
- 3: *Maintain the Vision*
- 4: *What You Produce, Others Will Consume*
- 5: *Be Open to the Future*
- 6: *Plan Ahead for Reuse*
- 7: *Think!*

# The Cost of Software Production




Relative costs of the stages of software development

# Why Design is Important?

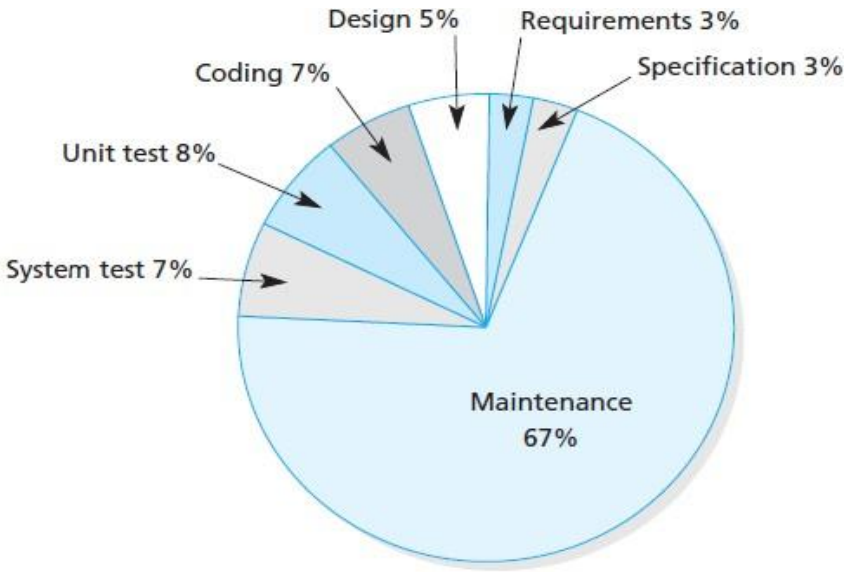



Relative numbers of errors made during the stages of software development

# Maintenance

- Maintenance is the term for any effort that is put into a piece of software after it has been written and put into operation. There are two main types:
  - ***Remedial(Corrective) maintenance***, which is the time spent correcting faults in the software (fixing bugs)
  - ***Adaptive maintenance***, which is modifying software either because the users' needs have changed or because, for example, the computer, operating system or programming language has changed

# Maintenance




Relative costs of the stages of software development