Software Engineering

Abdullah Al Shiam

Lecturer

Dept. Of CSE

Sheikh Hasina University, Netrokona

Recommended Books-

- **Software Engineering, A Practitioner's Approach** (6th / 7th Edition) by **ROGER S. PRESSMAN**
- **Software Engineering** by **Ian Sommerville, Addison-Wesley**

An Introduction to Software Engineering

What is Software?

- Software is
  - **instructions** (computer programs) that when executed provide desired function and performance,
  - **data structures** that enable the programs to adequately manipulate information, and
  - **documents** that describe the operation and use of the programs.

Dual Role of Software

- Software is a **product**
  - -Delivers computing potential (capability)
  - -Produces, manages, acquires, modifies, displays, or transmits information
- Software is a **vehicle** (medium) 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)

What is Software Engineering?

If you don't understand it, you can't program it.
If you didn't measure it, you didn't do it.

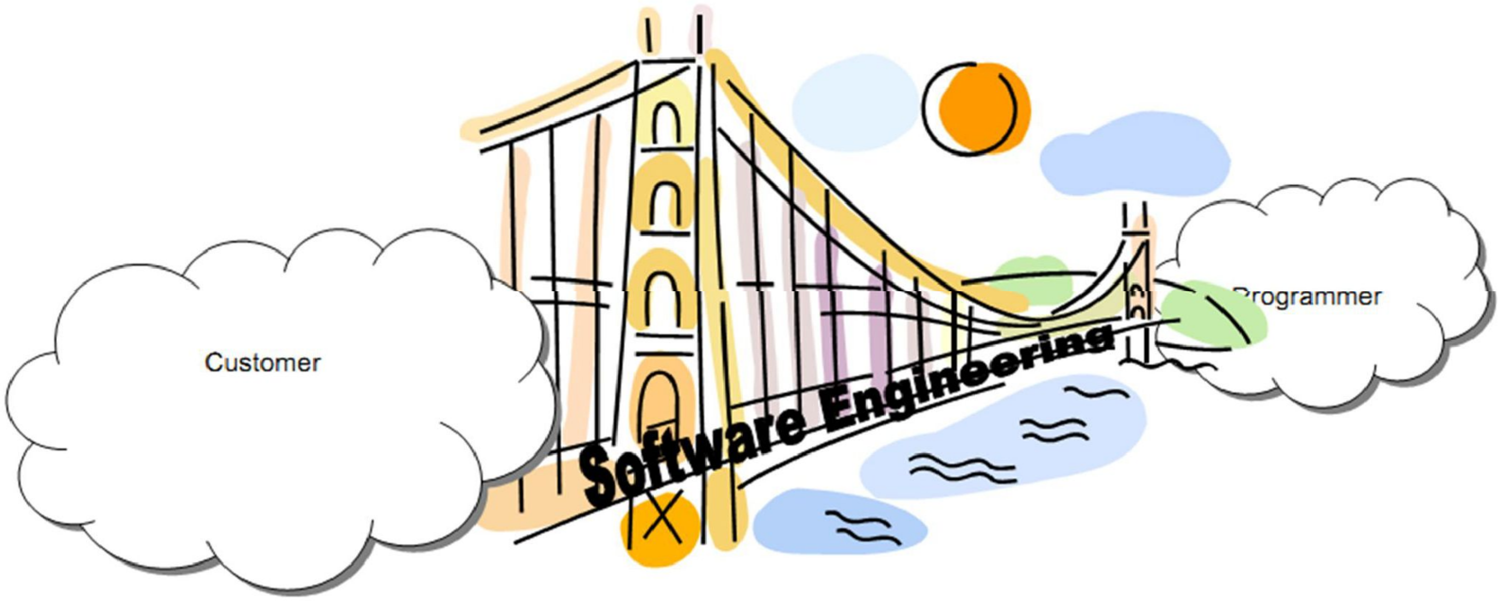
“Software Engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software.”

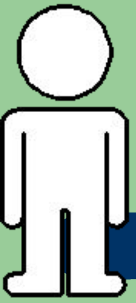
-IEEE’90

“Software Engineering is an act of applying 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.”

The Role of Software Engg.

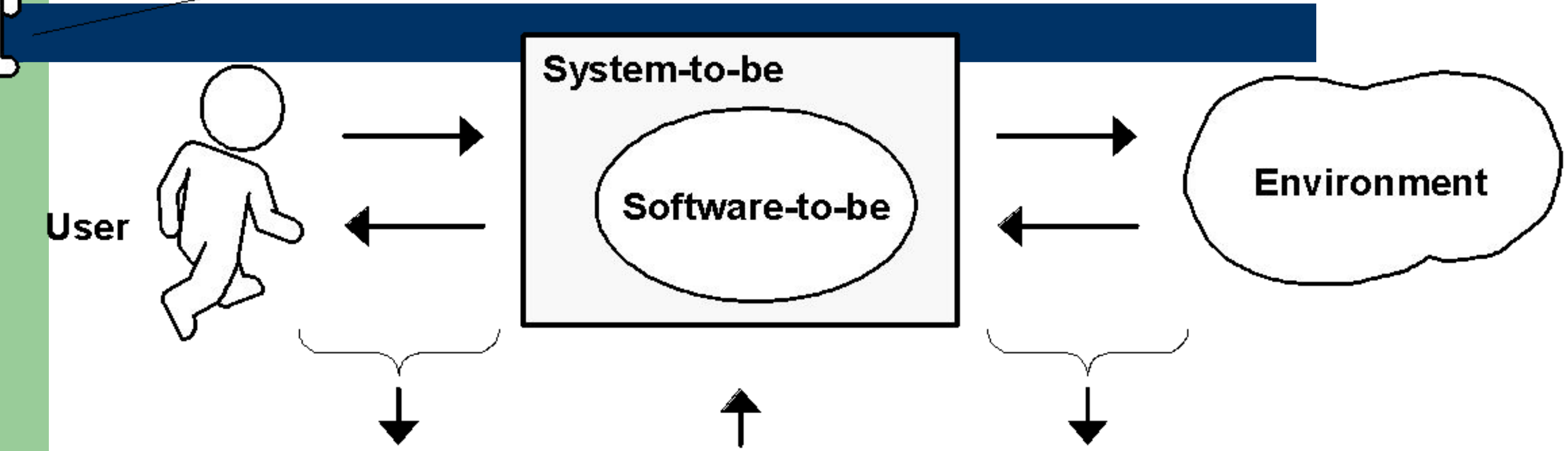
The role of software engineering is to capture the customer's **business needs** and specify the **"blueprints"** for the system so that programmers can implement it.





Customer:

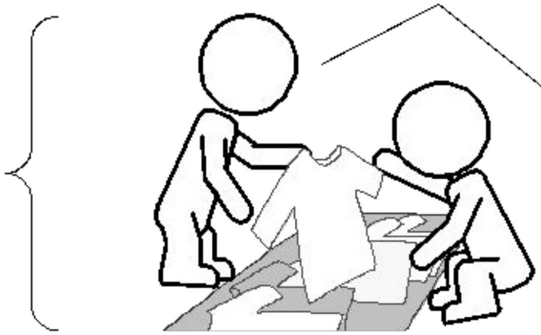
Requires a computer system to *achieve some business goals* by user interaction or interaction with the environment in a specified manner



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

May be the same person



Programmer's task:

To ***implement*** the software-to-be designed by the software engineer

Objectives of Software Engineering

- Understanding user conceptual model & development of better specification.
- Improvement in design languages & reusable code.
- Specification of interface & mockup to confirm specification.
- Improving the quality of the software products.
- Increasing the productivity & Giving job satisfaction to the software engineers.

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

What is the difference between software engineering and system engineering?

System engineering	Software engineering
System engineering deals with all aspects of computer-based system development including hardware, software and process engineering.	Software engineering is a part of system engineering.
System engineers are involved in system specification, architectural design, integration and deployment.	Software engineering is to tell the practicalities of developing and delivering useful software.

Software Characteristics

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

Hardware vs. Software

- Hardware
  - Manufactured
  - Wear out
  - Built using components
  - Relatively simple

- Software
  - Developed / engineered
  - Deteriorate/Obsolete
  - Custom built
  - Complex

Software is developed or engineered, it is not manufactured

- Some similarities exist between software development and hardware manufacture,

  -the two activities are fundamentally different.

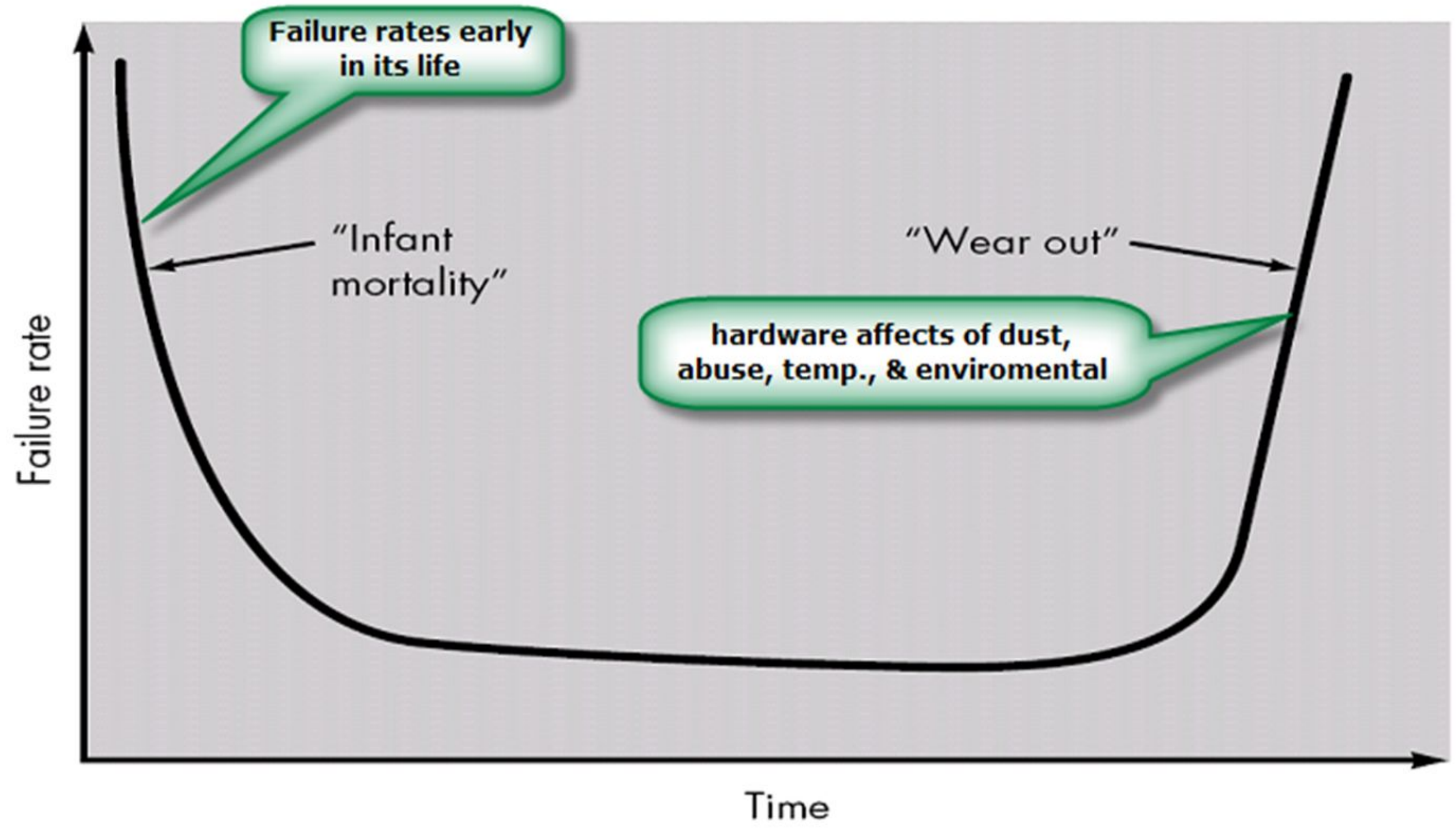
- High quality is achieved through good design, ⇨ in both case

- Once a hardware product has been manufactured, it is difficult or impossible to modify.

Software is developed or engineered, it is not manufactured

- In contrast, software products are routinely modified and upgraded.
- In hardware, hiring more people allows you to accomplish more work, but the same does not necessarily hold true in software engineering.

Software doesn't "wear out”



Software doesn't "wear out”

- As time progresses, the hardware components start deteriorating

  -they are subjected to environmental maladies such as dust, vibration, temperature etc. and at some point of time they tend to breakdown.

- The defected components can then be traced and replaced .

Software doesn't "wear out”

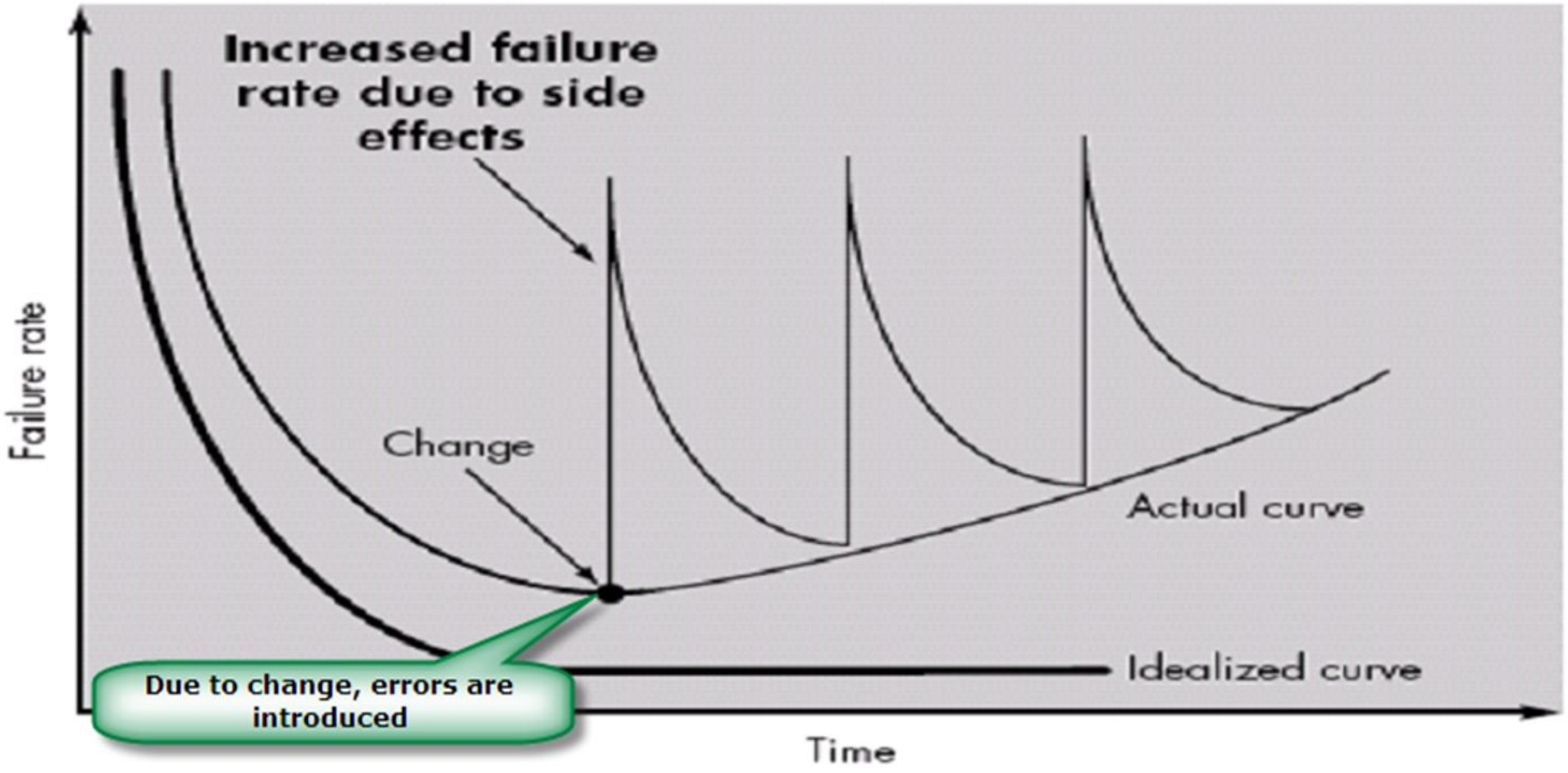
- But, software is not susceptible to the environmental changes .

  -So Do, it does not wear out.

- The software works exactly the same way even after years it was first developed unless any charges are introduced to it.

- The changes in the software may occur due to the changes in the requirements .And these changes may introduce some defects in it thus, deteriorating the quality of software .So, software need to maintained properly.

Software doesn't "wear out”



Software continues to be custom built

- Most of the engineered products are first designed before they are manufactured.
- Designing includes identifying various components for the product before they are actually assembled.
- Here several people can work independently on these components thus making the manufacturing system highly flexible.

Software continues to be custom built

- In software, breading a program into modules is difficult task , since each module is highly interlinked with other modules.
- Further, it requires lot of skill to integrate different modules into one .
- Now a days the term component is widely used in software industry where object oriented system is in use.

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 acceptable.

- **Maintainability**
  - Software must evolve to meet changing needs;
- **Dependability**
  - Software must be trustworthy;

What are the attributes of good software?

- **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.

Software change

Software change is inevitable:

- New requirements emerge when the software is used;
- The business environment changes.
- Discovered errors must be repaired.
- New computers and equipment are added to the system.
- The performance or reliability of the system may need to be improved.

Software Evolution

- Software evolution is the term to refer to the process of developing software initially, then repeatedly updating it for various reasons.

Importance of evolution

- Organizations have huge investments in their software systems - they are critical business assets.
- To maintain the value of these assets to the business, they must be changed and updated.
- The majority of the software budget in large companies is devoted to evolving existing software rather than developing new software.

Software crisis-Problem

The problem list included software that was

- Unreliable
- Delivered late
- Prohibitive in terms of modification costs
- Impossible to maintain
- Performing at an inadequate level
- Exceeding budget costs

.

Software crisis-Problem

The crisis manifested itself in several ways:

- Projects running over-budget.
- Projects running over-time.
- Software was vary inefficient.
- Software was of low quality.
- Software often did not meet requirements.
- Projects were unmanageable and code difficult to maintain.
- Software was never delivered

Software crisis Cause

- Bad planning or run time decisions.
- Lack of documentation.
- Improper assessment.
- Incorrect estimates of needed resources.
- Impractical project goals.
- Not a good communication among customers, developers and users.
- Use of undeveloped technology.
- Not a good project management.

Software Applications

- **System software**: such as compilers, editors, file management utilities.
- **Application software**: stand-alone programs for specific needs.
- **Real-time software:** Software that monitors/analyzes/controls real-world events as they occur is called real time.
- **Personal computer software**: The personal computer software market has burgeoned over the past two decades. Word processing, spreadsheets, computer graphics, multimedia, entertainment, database management.
- **Engineering/scientific software**: Characterized by “number crunching “algorithms such as automotive stress analysis, molecular biology, orbital dynamics etc.

Software Applications

- **Web-based software**: The Web pages retrieved by a browser are software that incorporates executable instructions (e.g., CGI, HTML, Perl, or Java), and data (e.g.,One of the most comprehensive libraries of shareware/freeware can be found at hypertext and a variety of visual and audio formats).
- **Embedded software** resides within a product or system. (key pad control of a microwave oven, digital function of dashboard display in a car).
- **Artificial intelligence** software uses non-numerical algorithm to solve complex problem. Robotics, expert system, pattern recognition game playing.