Introduction to Software Engineering

Slide Set - 2

Organized & Presented By:
Software Engineering Team
CSED
TIET, Patiala

What is software engineering?

Software engineering is an engineering discipline which is concerned with all aspects of software production

Software engineers should

- adopt a systematic and organised approach to their work
- use appropriate tools and techniques depending on
  - the problem to be solved,
  - the development constraints and
  - the resources available

SE history

- SE introduced first in 1968 – conference about "software crisis" when the introduction of third generation computer hardware led more complex software systems then before
- Early approaches based on informal methodologies leading to
  - Delays in software delivery
  - Higher costs than initially estimated
  - Unreliable, difficult to maintain software
- Need for new methods and techniques to manage the production of complex software.

Why Software Engineering?

- **Software development is hard** !
- **Important to distinguish "easy" systems** (*one developer, one user, experimental use only*) **from "hard" systems** (*multiple developers, multiple users, products*)
- **Experience with "easy" systems is misleading**
  - *One person techniques do not scale up*
- **Analogy with bridge building:**
  - Over a stream = easy, one person job
  - Over River Ganga … ? (*the techniques do not scale)*

Why Software Engineering ?

- The problem is ***complexity***
- Many sources, but ***size*** is key:
  - UNIX contains 4 million lines of code
  - Windows 2000 contains 10^8 lines of code

Software engineering is about managing this complexity.

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.

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 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

What are the **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 Characteristics

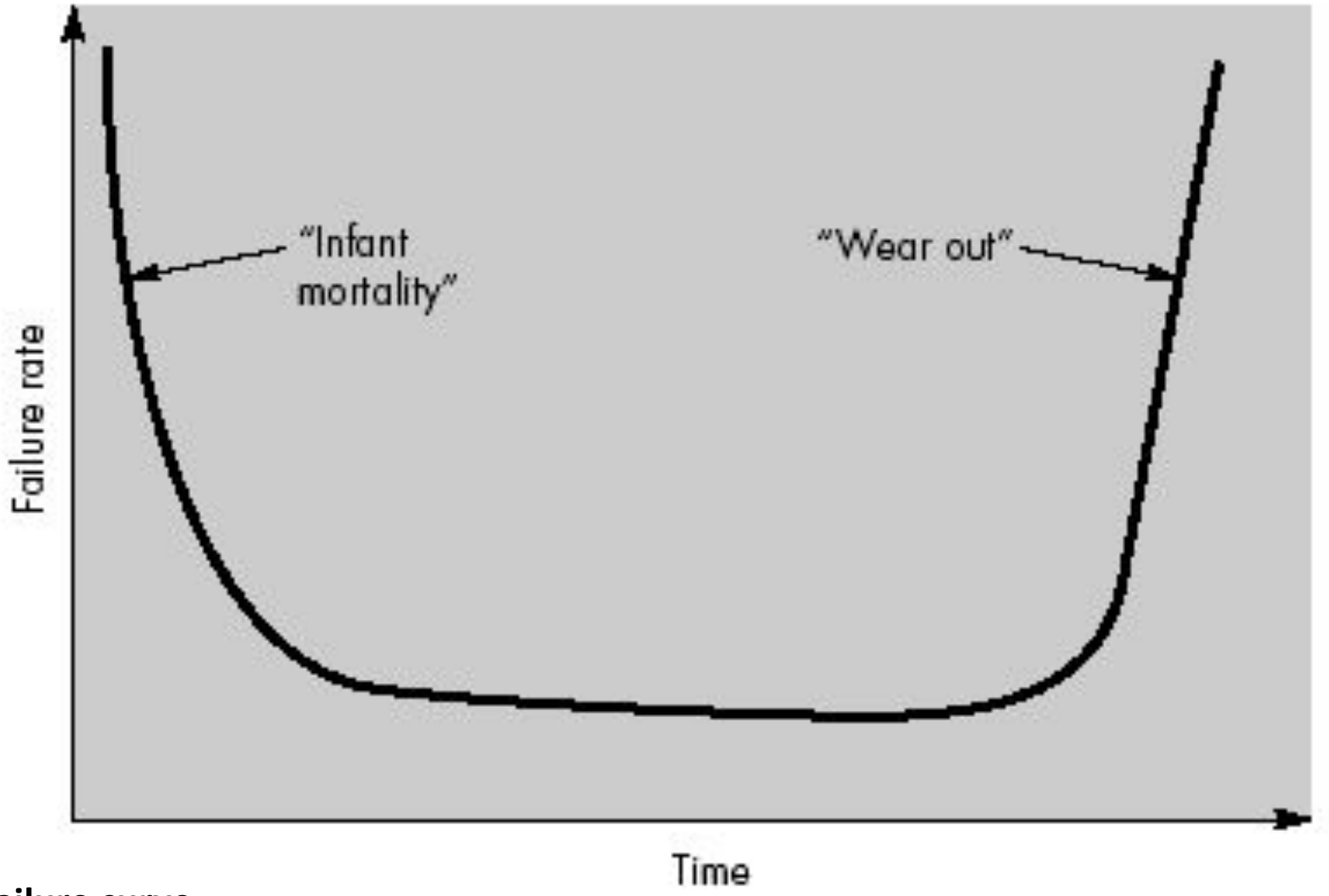
- To gain an understanding of software (and ultimately an understanding of software engineering)
- it is important to examine the characteristics of software that make it different from other things that human beings build.

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.

- Both activities are dependent on people.
- but the relationship between people applied and work accomplished is entirely different
- Software costs are concentrated in engineering.
- This means that software projects cannot be managed as if they were manufacturing projects.

Software doesn't "wear out."



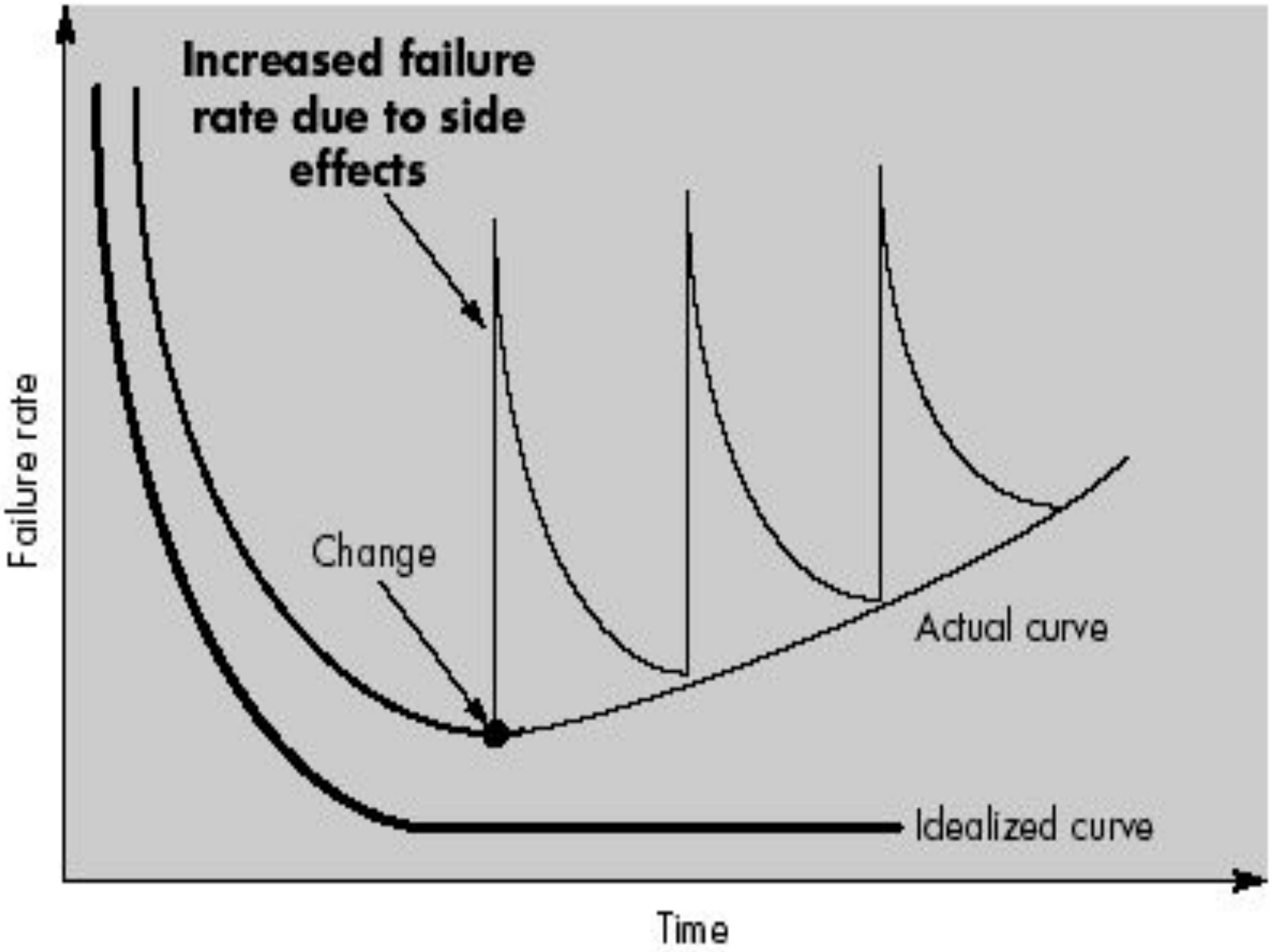
Failure curve for hardware

- Figure depicts failure rate as a function of time for hardware.
- The relationship, often called the "bathtub curve," indicates that hardware exhibits relatively high failure rates early in its life
- these failures are often attributable to design or manufacturing defects.
- defects are corrected and the failure rate drops to a steady-state level (ideally, quite low) for some period of time.
- As time passes, however, the failure rate rises again

- hardware components suffer from the cumulative affects of dust, vibration, abuse, temperature extremes, and many other environmental maladies.

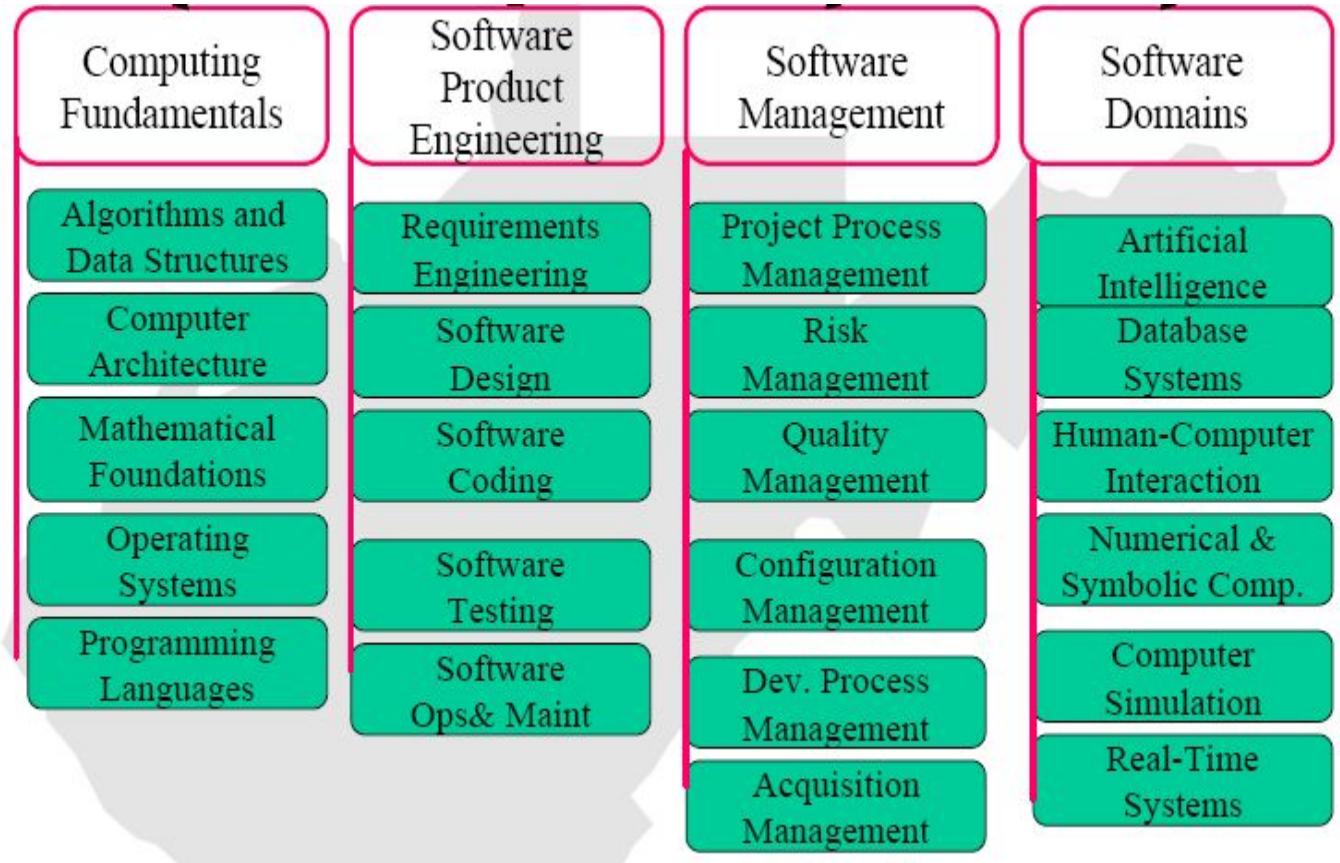
- Stated simply, *the hardware begins to wear out.*

- Software is not susceptible to the environmental maladies that cause hardware to wear out.
- In theory, therefore, the failure rate curve for software should take the form of the “idealized curve”
- Undiscovered defects will cause high failure rates early in the life of a program.
- However, these are corrected (ideally, without introducing other errors) and the curve flattens as shown.
- The idealized curve is a gross oversimplification of actual failure models for software. However, the implication is clear software doesn't wear out. But it does deteriorate!


Increased failure rate due to side effects
Failure rate
Change
Actual curve
Idealized curve
Time

- Another aspect of wear illustrates the difference between hardware and software.
- When a hardware component wears out, it is replaced by a spare part.
- There are no software spare parts.
- Every software failure indicates an error in design or in the process through which design was translated into machine executable code.

Software Engineering Body of Knowledge



Source: http://www.sei.cmu.edu/pub/documents/99.reports/pdf/99tr004.pdf

Software Applications

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)
- Other systems applications (e.g., operating system components, drivers, telecommunications the system software area is characterized by heavy interaction with computer hardware;

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.

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.

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.

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.

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., hypertext and a variety of visual and audio formats).
- In essence, the network becomes a massive computer providing an almost unlimited software resource that can be accessed by anyone with a modem.

Artificial intelligence software.

- Artificial Intelligence (AI) software makes use of non-numerical algorithms to solve complex problems that are not amenable to computation or straightforward analysis.
- Expert systems, also called knowledge based systems,
- Pattern recognition (image and voice), artificial neural networks, theorem proving, and game playing are representative of applications within this category.