Lecture – 06
Software Engineering

Section –A

Outline of the talk

- Evolving role of software
- Emergence of Software Engineering
- Software characteristics
- Software Applications
- Software Products
- Software Crisis
- Emergence of Software Engineering

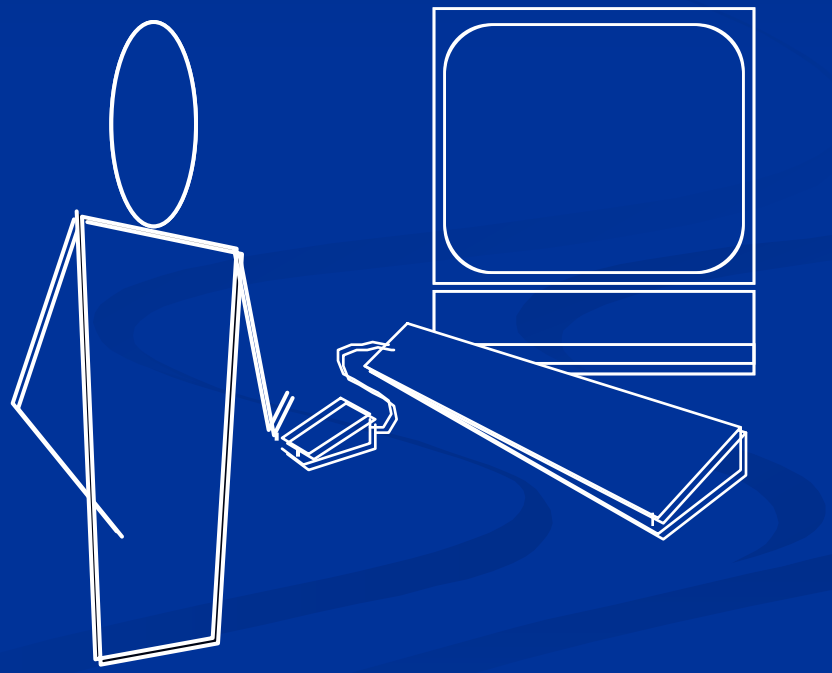
Evolving Role of Software : Software's Dual Role

- Software is a product
  - Delivers computing potential embodied by computer hardware.
  - whether S/W resides in cellular phone or computer its an information transformer.
  - Produces, manages, acquires, modifies, displays, or transmits information, that information can be as simple as single bit or as complex as multimedia presentation.
- 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)

What is Software?

Software is a set of items or objects that form a “configuration” that includes

- Computer programs(instructions)
- Documents that describes operation and use of programs (software manuals)
- Data structures that enable the programs to manipulate information ...



Software Characteristics

- **Software is developed or engineered** :

✓ High quality is achieved through good design.

✓ Requires construction, software costs are concentrated in development.

✓ Quality problems are non-existent.

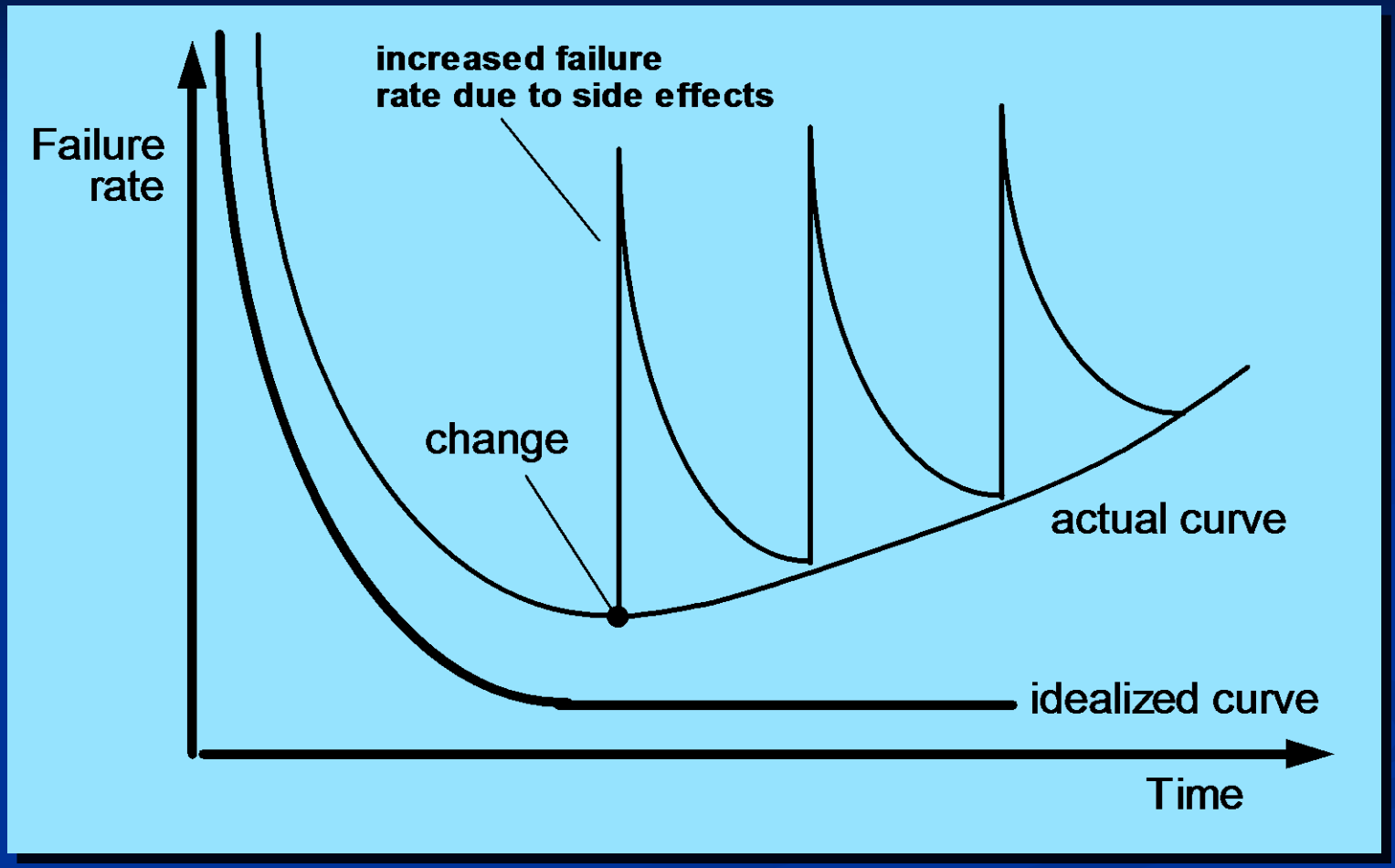
- **software doesn't wear out:**

➢ Software is not liable to environmental maladies(problems) that cause hardware to wear out. Therefore, Failure rate curve for software should take the form of "idealized curve"

➢ Undiscovered defects will cause high failure rates early in the life of a program, these are corrected(hopefully without introducing other errors) and the curve flattens as shown.

Implication is clear :Software doesn't wear out but it does deteriorate !

Wear vs. Deterioration(decline)



Contradiction can be best explained by considering the “actual curve”

- During its life, software will undergo change. As changes are made, it is likely that errors will be introduced, causing the failure rate to spike as shown in fig.
- Before the curve can return to the original steady state failure rate, another change is requested, causing the curve to spike again.
- Slowly, the minimum failure rate level begins to rise – the software is deteriorating (failing/weakening) due to change.

- **Most software is custom-built, rather than being assembled from existing components.**
- A software component should be designed and implemented so that it can be reused in many different programs.
- Modern reusable components summarize both data and processing that is applied to the data, enabling the s/w engineer to create new applications from reusable parts.

Software Applications

- **System software**

A collection of programs written to service other programs at system level. It is characterized by

- heavy interaction with computer hardware
- heavy usage by multiple users.
- concurrent operation that requires scheduling, resource sharing , process management

For example, compiler, operating systems.

- **Application software**

✓ it consists of standalone programs that solve a specific business need.

✓ applications in this area process business or technical data in a way that facilitates business operations or management/technical decisions making.

Eg.: point of scale transaction, real-time manufacturing process control.

- **Engineering/scientific software** **applications** range from:
  - ✓ astronomy to volcanology
  - ✓ from automotive stress analysis to space shuttle orbital dynamics.
  - ✓ From molecular biology to automated manufacturing.
- **Embedded software**
  - ➢ It 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.

  Eg. : digital functions in an automobile such as fuel control, dashboard displays, braking systems etc.

- **WebApps (Web applications)**
  - "WebApps", span a wide array of applications.
  - It can be little more than a set of linked hypertext files that present information using text and limited graphics.
  - "WebApps" are evolving into sophisticated computing environments that not only provide standalone features, computing functions, and content to the end-user, but also are integrated with corporate databases and business applications.
- **AI software**

Programs make use of AI techniques and methods to solve complex problems.

Eg. Active areas within this area include robotics, expert systems, pattern recognition (image and voice), artificial neural networks and games

Software Products

A software product is typically a single application or suite of applications built by a software company to be used by *many* customers, businesses or consumers.

Two types are there:

- Generic Products
- Customized Products

- Generic software development is the one done for "General Purpose" audience while "Custom Software Development" is done to satisfy a particular need of a particular client.
- General Purpose software development is tough as compared with Custom made; not by the skills required to develop the application but by the design and marketing point of view.

- **Generic Products:**
  - In General Purpose application/product design and development, you will need to "imagine" what an end-user require. Here, the term "end - user" has no face; you have to imagine it.
  - Market Surveys and general Customer Demand analysis may help a company to reduce the risk factor and think about some innovations over existing similar solutions

- The classic example for a generic software product would be something like an operating system (Mac OS X) or a word processor (Microsoft Word).
- As a rule, most users want the same sorts of functionality out of such a thing, and it would be rare for a company to decided to build it’s own word processor to meet it’s own specific requirements.

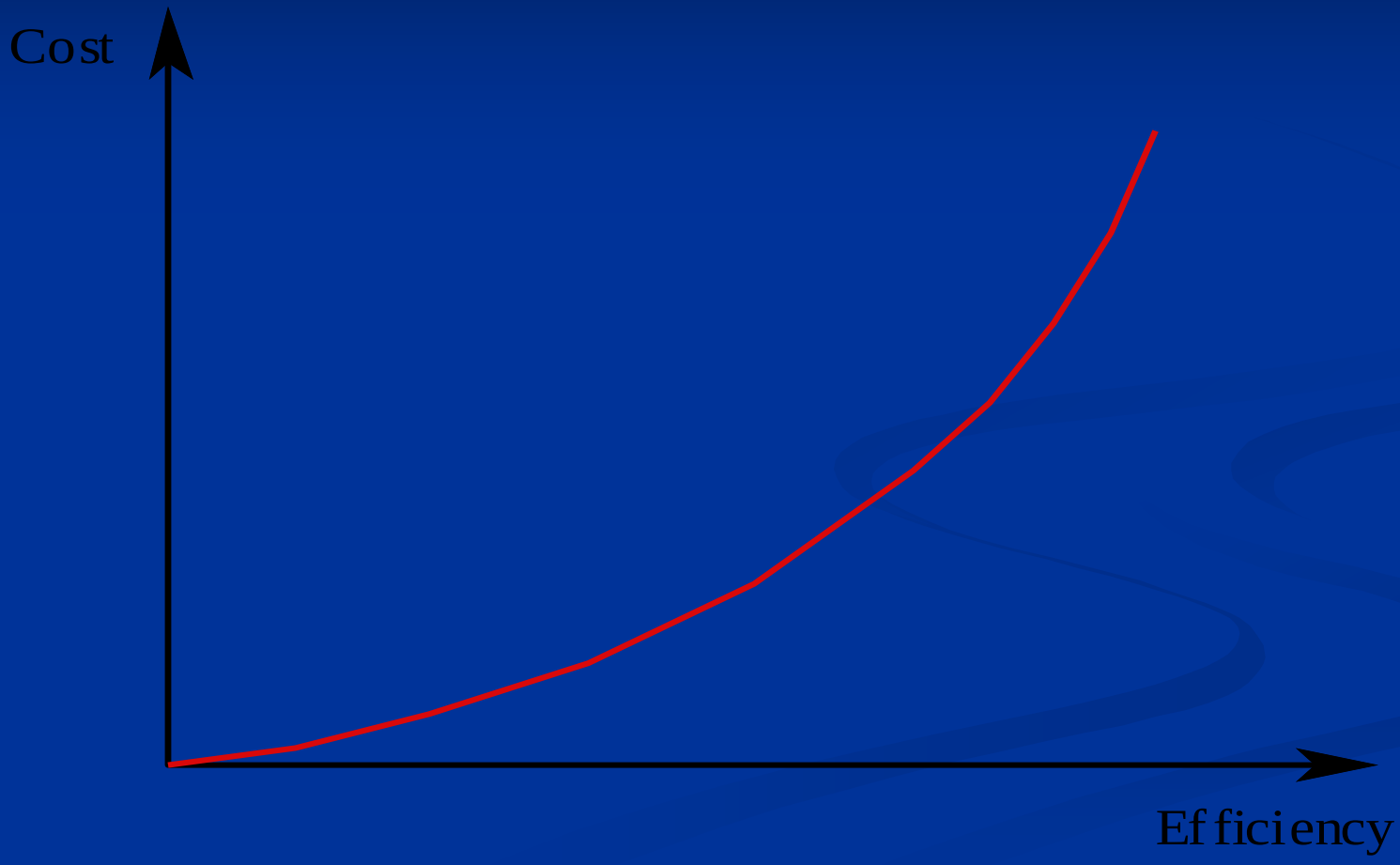
- **Customized Products:**
  - "Custom Software product Development" is done to satisfy a particular need of a particular client.
  - Custom Made application/product development, you have a specific face of "end - user" in front of you. You know whom you need to satisfy.
  - Understanding the need and Analyzing it to get the best out of it is a ***challenge*** here.
  - Planning and reaching the goals within dead-line adds a value to your software development excellence as a **professional service provider.**

- By Buyer's (or Client's) point of view, now a days it's preferred to have a Custom Made application developed rather than buying a General Purpose software.
- You get the application done that exactly matches your requirements and also most importantly your style of computing.

Software Product Attributes

- Maintainability : Providing Maintenance to the products.
- Dependability : Can be on h/w , s/w platforms
- Efficiency : w.r.t. cost.
- Usability ,reusablity.

Efficiency Costs



Software Crisis

- **Software crisis** was a term used in the early days of computing science. The term was used to describe the impact of rapid increases in the power of computers and the complexity of the problems which could be tackled.
- In essence, it refers to the difficulty of writing correct, understandable, and verifiable computer programs.
- The roots of the software crisis are complexity, expectations, and change.
- The causes of the software crisis were linked to the overall complexity of hardware and the software development process.

The crisis manifested itself in several ways:

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

Note :These are just hints, question can be asked for 10 or 20 marks. . For that you need to elaborate these points with some examples wherever required.

- Various processes and methodologies have been developed over the last few decades to "tame" the software crisis, with varying degrees of success.
- However, it is widely agreed that there is no "silver bullet" — that is, no single approach which will prevent project overruns and failures in all cases.
- In general, software projects which are large, complicated, poorly-specified, and involve unfamiliar aspects, are still particularly exposed to large, unexpected problems.

Emergence of Software Engineering

What is Software Engineering ???

The study of systematic and effective processes and technologies for supporting software development and maintenance activities

- Improve quality
- Reduce costs

Why is software engineering needed?

- To predict time, effort, and cost
- To improve software quality
- To improve maintainability
- To meet increasing demands
- To lower software costs
- To successfully build large, complex software systems
- To facilitate group effort in developing software

Who Needs Software Engineering?

Show me a business in the U.S. that doesn't use Software

So…..

Everyone !

Historical Perspective

- 1940s: computers invented
- 1950s: assembly language, Fortran
- 1960s: COBOL, ALGOL, PL/1, operating systems
- 1969: First conference on Software Eng
- 1970s: multi-user systems, databases, structured programming
- 1980s: networking, personal computing, embedded systems, parallel architectures
- 1990s: information superhighway, distributed systems, OO in widespread use.
- 2000s: virtual reality, voice recognition, video conferencing, global computing, ...

Assignment

- Define software engineering.
- What is software crisis.