SOFTWARE ENGINEERING
CSE-305

Books

References:

- 1. Roger S Pressman, “Software Engineering: A Practitioners Approach”, McGrawHill Publications, 6th Edition.
- 2. Hans Van Valiet, ”Software Engineering: Principles and Practice”, Wiley India, 3rd Edition.

Learning outcomes!!!

- Introduction to Software Engineering
- The Evolving Role of Software
- The changing nature of software
- Legacy software
- Software Myths
- Generic View of Process

Introduction to S/W Engg.

- “The law of unintended consequences” – invention of one technology can have pro-found and un expected effects on other seemingly unrelated technologies, on commercial enterprises, on peoples and even on cultures as a whole.
- Software is one such
- Infused 1. Genetic Engineering 2.Telecommunications 3.Printing technology and so on so forth..

Software Engineering!!!

- The framework encompasses a process, a set of methods, and an array of tools called software engineering.
- Website design and implementation
- Object Oriented Systems or Aspect Oriented Programming
- Others - Linux

Evolving role of Software!!!

- Software takes a dual role today – as a product and a vehicle for delivering a product.
- As a product, it delivers the computing potential embodied by computer hardware or more broadly, by a network of computers that can be accessible by local hardware.
- As a vehicle, software acts as the basis for control of the computer (OS), the communication of information (N/W's) and the creation and control of other programs (S/W tools and environments)

Software

What is Software?

- Software is the product that software professional build and support over long term.
- Software is combination of set of instructions and accompanying documents.

What does a S/W do???

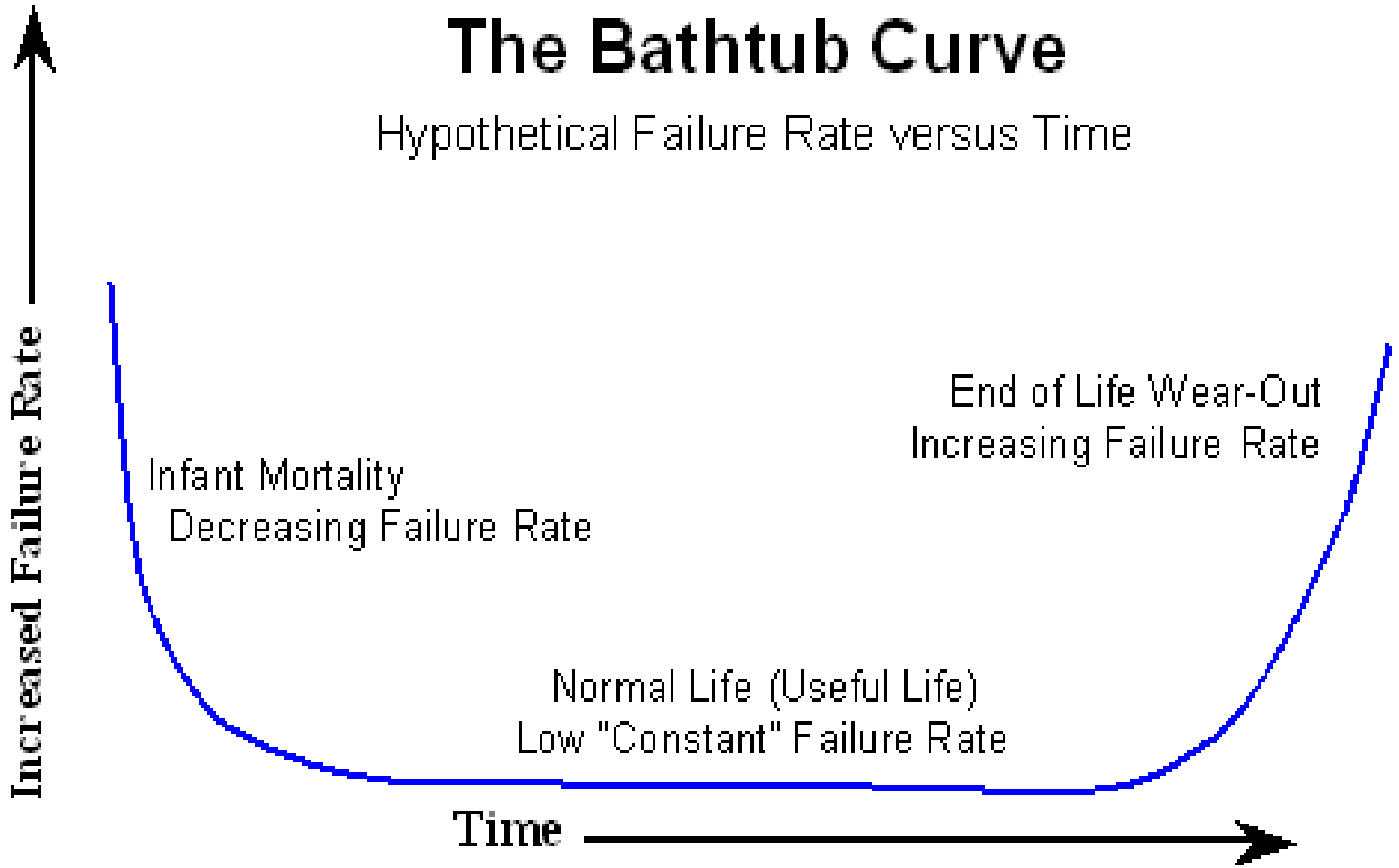
The software

- Provides good product with useful information
- Transforms the data so that it can be more useful in a local context
- Manages business information to enhance competitiveness
- Provides a gateway to worldwide networks like internet

Software Characteristics

- Software is a logical related rather than a physical system.
- Software is developed or engineered, it is not manufactured in the Classical Sense.
- Software does not wear out.
- Although the industry is moving toward component-based assembly, most software continues to be custom built.

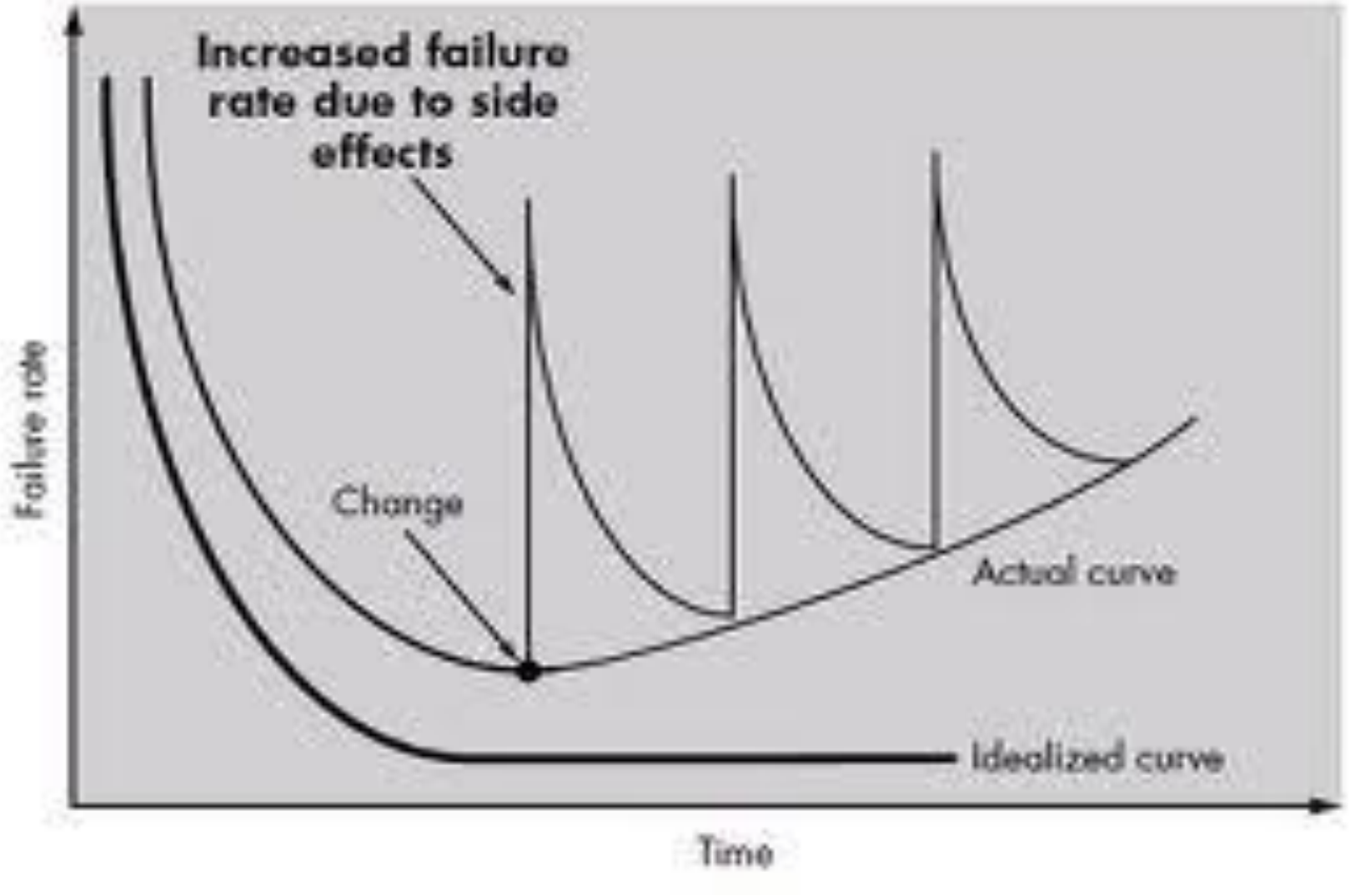
Failure curve for hardware



H/W – Bath tub curve

- Period A, B and C stands for burn-in phase, useful life phase and end-of-life phase.

Failure curve for software



S/W Curve

- Software reliability, however, does not show the same characteristics similar as hardware.
- There are two major differences between hardware and software curves. One difference is that in the last phase, software does not have an increasing failure rate as hardware does. In this phase, software is approaching obsolescence; there are no motivation for any upgrades or changes to the software. Therefore, the failure rate will not change.
- The second difference is that in the useful-life phase, software will experience a drastic increase in failure rate each time an upgrade is made. The failure rate levels off gradually, partly because of the defects found and fixed after the upgrades.

Software failure mechanisms

- Software failures may be due to errors, ambiguities, oversights or misinterpretation of the specification that the software is supposed to satisfy, carelessness or incompetence in writing code, inadequate testing, incorrect or unexpected usage of the software or other unforeseen problems.

characteristics of software compared to hardware

- **Failure cause**: Software defects are mainly design defects.
- **Wear-out**: Software does not have energy related wear-out phase. Errors can occur without warning.
- **Repairable system concept**: Periodic restarts can help fix software problems.
- **Time dependency and life cycle**: Software reliability is not a function of operational time.
- **Environmental factors**: Do not affect Software reliability, except it might affect program inputs.
- **Reliability prediction**: Software reliability can not be predicted from any physical basis, since it depends completely on human factors in design.

characteristics of software compared to hardware

- **Redundancy**: Can not improve Software reliability if identical software components are used.
- **Interfaces**: Software interfaces are purely conceptual other than visual.
- **Failure rate motivators**: Usually not predictable from analyses of separate statements.
- **Built with standard components**: Well-understood and extensively-tested standard parts will help improve maintainability and reliability. But in software industry, we have not observed this trend. Code reuse has been around for some time, but to a very limited extent. Strictly speaking there are no standard parts for software, except some standardized logic structures.

Types of Software

- System Software (Compilers, Editors, Operating System, File Management Utilities, Device drivers, network and telecom software)
- Application Software (Data Processing software, Banking software)
- Scientific software (Astronomy to volcanology, CAD, system simulation)

Types of Software(continued)

- Embedded software (Oven, Washing Machine, Elevator)
- Product Line S/W (Entertainment, CG, Multimedia)
- Web Applications (B2B, B2C)
- Artificial Intelligence (Robotics, Expert Systems, Pattern Recognition, ANN, Theorem Proving and Game Playing)
- Ubiquitous computing
- Netsourcing

Legacy software

- A **legacy system** is an old method, technology, computer system, or application program.
- Software systems that are developed specially for an organisation have a long lifetime
- Many software systems that are still in use were developed many years ago using technologies that are now obsolete
- These systems are still business critical that is, they are essential for the normal functioning of the business
- They have been given the name legacy systems

!!!!!

- It is common for a young software engineers to work on a software that is older than she is!!!
- Adding people to a late software project makes it later!!!

Define Software Engineering.

- Software Engineering is Systematic, Disciplined, Quantifiable approach for Development, Operation and Maintenance of Software.

Software Development Myths

- Pressman (1997) describes a number of common beliefs or myths that software managers, customers, and developers believe falsely.
- He describes these myths as ``misleading attitudes that have caused serious problems.''
- We look at these myths to see why they are false, and why they lead to trouble.

Software Customer Myths

- *A general statement of objectives is sufficient to begin writing programs—we can fill in the details later.*

- *Software requirements continually change, but change can be easily accommodated because software is flexible.*

(Refer text for reality behind these myths)

Developer Myths

- *The job is done when the code is delivered.*
- *Until I get the program "running" I have no way of assessing its quality.*
- *The only deliverable work product for a successful project is the working program.*
- *Software engineering will make us create voluminous and unnecessary documentation and will invariably slow us down.*

Software Management Myths.

- *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?*
  - . Standards have been developed by companies and standards organizations. They can be very useful. However, they are frequently ignored by developers because they are irrelevant and incomplete, and sometimes incomprehensible.
- *If we get behind schedule, we can add more programmers and catch up*
  - This solution seems intuitive: if there is too much work for the current team, just enlarge it. Unfortunately, increasing team size increases communication overhead. New workers must learn project details taking up the time of those who are already immersed in the project. Also, a larger team has many more communication links, which slows progress.
- *If I decide to outsource the software project to a third party, I can just relax and let that firm build it.*

References

- Fundamentals of Software Engineering 2nd Edition by Rajib Mall
- Software Engineering a practitioners Approach by Pressman R. S.
- Software Engineering Theory and Practice by Pfleeger S. L. and Joanne M. Atlee
- An Integrated Approach to Software Engineering by Pankaj Jalote