Software Engineering Unit-1

By

Mayurkumar Marolia

Software

- IEEE defines ***software*** as the collection of computer programs, procedures, rules, and associated documentation and data.

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

Software

- This definition clearly states that software is not just programs, but includes all the associated documentation and data.
- This implies that the discipline dealing with the development of software should not deal only with developing programs,
- but with developing all the things that constitute software.

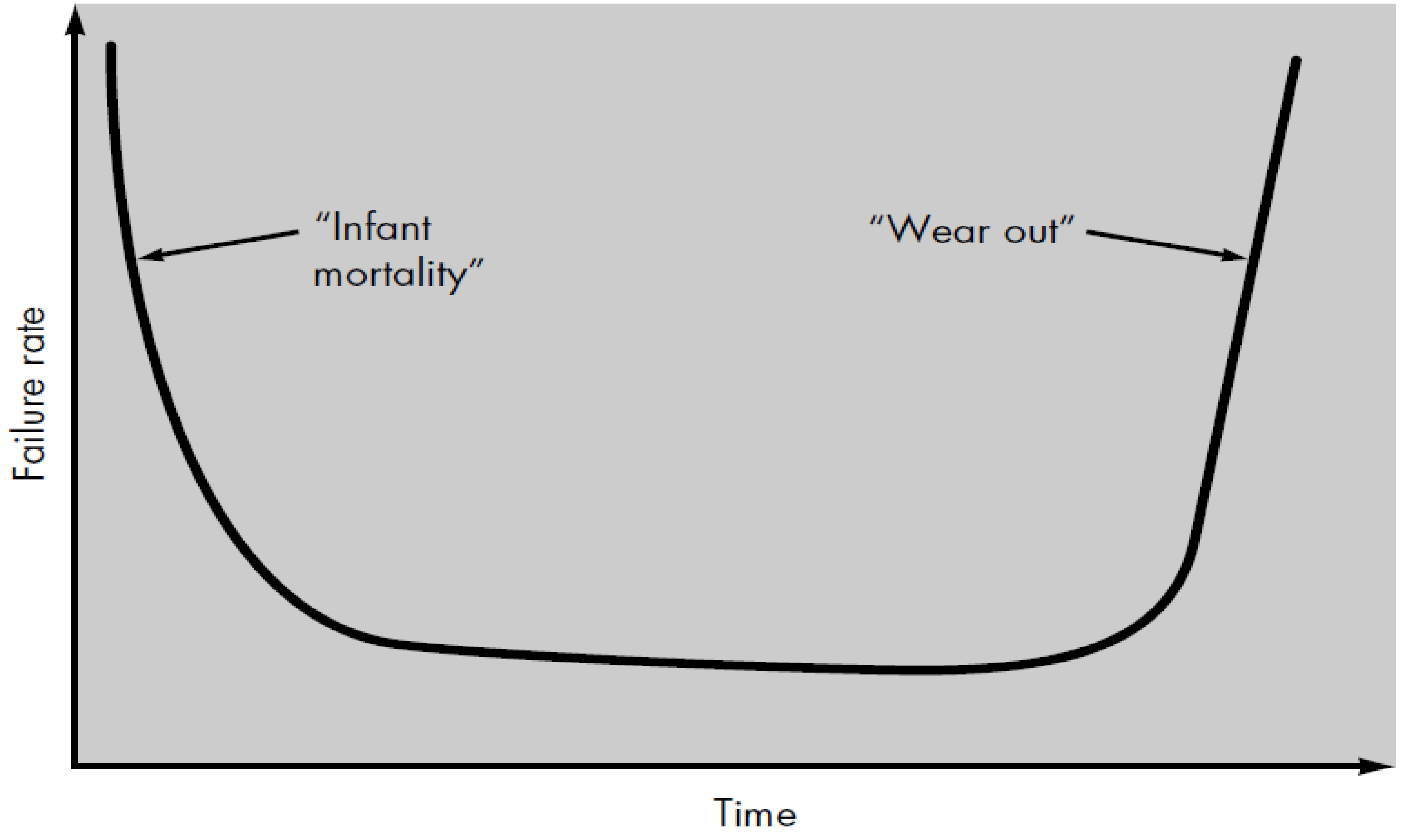
Software Characteristics

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

- 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 require the construction of a "product" but the approaches are different.

Software Characteristics

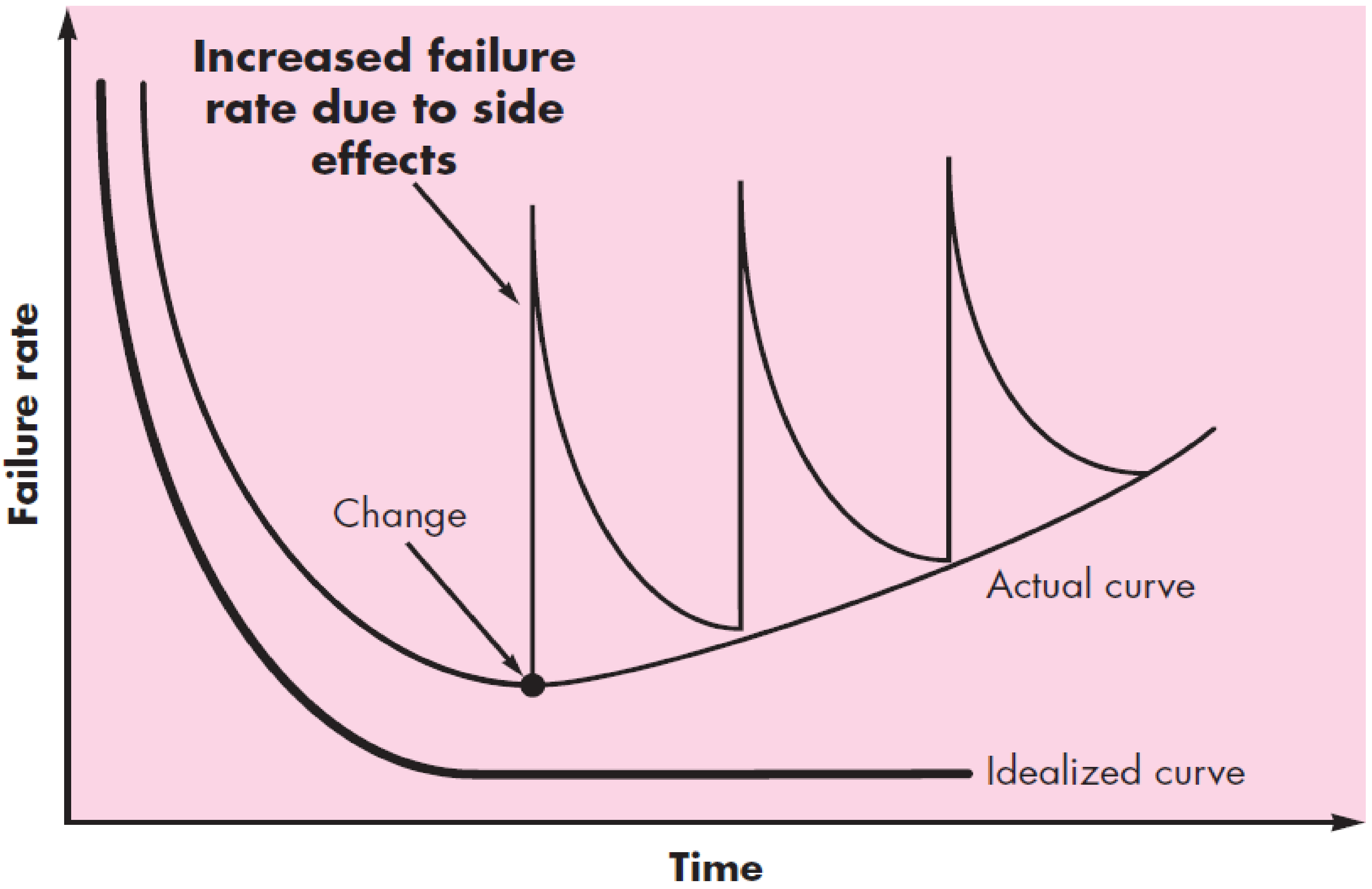
2. Software doesn't "wear out.“



Software doesn't "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***” shown in Figure .
- 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.
- However, the implication is clear—software doesn't wear out. But it does deteriorate!

Software Failure Curve



Software Characteristics

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

- In the hardware world, component reuse is a natural part of the engineering process. In the software world, it is something that has only begun to be achieved on a broad scale.
- software component should be designed and implemented so that it can be reused in many different programs.

Software Applications & Types

- Software may be applied in any situation for which a prespecified set of procedural steps (i.e., an algorithm) has been defined.
- System Software
- Real-Time Software
- Business Software
- Engineering & Scientific Software
- Embedded Software
- Personal Computer Software
- Web based Software
- Artificial Intelligence Software

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) process complex, but determinate, information structures. Other systems applications (e.g., operating system components, drivers, telecommunications processors) process largely indeterminate data.
- In either case, the system software area is characterized by
  - heavy interaction with computer hardware;
  - heavy usage by
  - multiple users;
  - concurrent operation that requires scheduling, resource sharing, and

    sophisticated process management; complex data structures; and multiple external interfaces.

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.

- Applications in this area restructure existing data in a way that facilitates business operations or management decision making.

- In addition to conventional data processing application, business software applications also encompass interactive computing (e.g., point of-sale transaction processing).

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.
- However, modern applications within the engineering/scientific area are moving away from conventional numerical algorithms.
- Computer-aided design, system simulation, and other interactive applications have begun to take on real-time and even system software characteristics.

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.
- Embedded software can perform very **limited and esoteric functions** (e.g., keypad control for a microwave oven) or provide significant function and control capability
- (e.g., digital functions in an automobile such as fuel control, dashboard displays, and braking systems).

Personal computer software

- 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 nonnumeric 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.

Software Engineering

- *Software engineering* is defined as the systematic approach to the development, operation, maintenance, and retirement of software.
- Software Engineering:
  - (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software;
  - that is, the application of engineering to software.
  - (2) The study of approaches as in (1).

Software Engineering & Problem Domain

- Problem Domain
  - Industrial Strength Software
  - Software is Expensive
  - Late and Unreliable
  - Maintenance and Rework

Industrial Strength Software

- Built to solve some problem of a client and is used by the clients organization for operating some part of business
- important activities depend on the correct functioning of the system. And a malfunction of such a system can have huge impact in terms of financial or business loss, inconvenience to users, or loss of property and life.
- Consequently, the software system needs to be of high quality with respect to properties like dependability, reliability, user-friendliness, etc.

Industrial Strength Software

- it requires that the software be **thoroughly tested** before being used.
- Building high quality software requires that the **development be broken into phases** such that output of each phase is evaluated and reviewed so bugs can be removed.

Software is Expensive

- Industrial strength software is very expensive primarily due to the fact that software development is extremely labor-intensive.
- To get an idea of the costs involved, let us consider the current state of practice in the industry.
- Lines of code (LOC) or thousands of lines of code (KLOC) delivered is by far the most commonly used measure of **software size** in the industry.
- As the **main cost of producing software** is the manpower employed, the cost of developing software is generally measured in terms of person-months of effort spent in development.
- productivity is frequently measured in the industry in terms of LOC (or KLOC) per person-month.

Late and Unreliable

- In a survey of over 600 firms, more than 35% reported having some computer-related development project that they categorized as a *runaway.* it is one where the budget and schedule are out of control.
- the software does not do what it is supposed to do or does something it is not supposed to do.
- In one defense survey, it was reported that more than 70% of all the equipment failures were due to software!
- this is in systems that are loaded with electrical, hydraulic, and mechanical systems.

Late and Unreliable

- In software, failures occur due to bugs or errors that get introduced during the design and development process.
- Hence, even though a software system may fail after operating correctly for some time, the bug that causes that failure was there from the start!
- It only got executed at the time of the failure.

Maintenance and Rework

- Why is maintenance needed for software?
  - because there are often some residual errors remaining in the system that must be removed as they are discovered.
- Once the software is delivered and deployed, it enters the *maintenance* phase.
- These errors, once discovered, need to be removed, leading to the software being changed. This is sometimes called *corrective maintenance.*

Maintenance and Rework

- Even without bugs, software frequently undergoes change.
- The main reason is that **software often must be upgraded and enhanced to include more features and provide more services.** This also requires modification of the software.
- It has been argued that once a software system is deployed, the **environment in which it operates changes.** Hence, the needs that initiated the **software development also change to reflect the needs of the new environment.**
- Hence, the software must adapt to the needs of the changed environment. The changed software then changes the environment, which in turn requires further change. This is phenomena is called ***adaptive maintenance.***

Phased Development Process

- A development process consists of various phases, each phase ending with a defined output.
- The phases are performed in an order specified by the process model being followed. (like Waterfall, Prototype, etc.)
- The main reason for having a phased process is that it breaks the problem of developing software into successfully performing a set of phases, each handling a different concern of software development.

Phased Development Process

- This ensures that the cost of development is lower than what it would have been if the whole problem was tackled together.
- Furthermore, a phased process **allows proper checking for quality and progress at some defined points** during the development (end of phases).
- Without this, one would have to wait until the end to see what software has been produced.
- Clearly, this will not work for large systems.
- Hence, for **managing the complexity, project tracking, and quality, all the development processes consist of a set of phases.**

Phased Development Process

- **Phased Development Process**
  1. **Requirements Analysis**
  2. **Software Design**
  3. **Coding**
  4. **Testing**

Requirements Analysis

- Requirements analysis is done in order to understand the problem the software system is to solve. The emphasis in requirements analysis is on identifying **what is needed** from the system, not how the system will achieve its goals.
- For complex systems, even determining what is needed is a difficult task.
- The **goal** of the requirements activity is to document the requirements in a *software requirements specification(SRS)* document.

Requirements Analysis

- There are two major activities in this phase:
  - problem understanding
  - analysis and requirement specification.

- In problem analysis, the aim is to understand the problem and its context, and the requirements of the new system that is to be developed.

Requirements Analysis

- Once the problem is analyzed and the essentials understood, the requirements must be specified in the requirement specification document (SRS).
- The requirements document must specify
  - **all functional and performance requirements**;
  - the **formats** of inputs and outputs; and
  - all **design constraints** that exist due to political, economic, environmental, and security reasons.

Software Design

- The purpose of the design phase is to *plan a solution of the problem* specified by the requirements document.
- This phase is the first step in moving from the problem domain to the solution domain.
- In other words, starting with *what* is needed, design takes us toward *how* to satisfy the needs.
- The design of a system is perhaps the most critical factor affecting the quality of the software;
- it has a major impact on the later phases, particularly testing and maintenance.

Software Design

- The design activity often results in three separate *outputs*
  - *Architecture design,*
  - *high level design,* and
  - *detailed design.*
- *Architecture* focuses on looking at a system as a combination of many different components, and how they interact with each other to produce the desired results.
- The *high level design* identifies the modules that should be built for developing the system and the specifications of these modules.
- At the end of system design all the major data structures, file formats, output formats, etc., are also fixed.
- In *detailed design,* the internal logic of each of the modules is specified.

Coding

- The **goal** of the coding phase is to **translate the design of the system into code** in a given programming language.
- For a given design, the aim in this phase is to implement the design in the best possible manner.
- The **coding phase affects both testing and maintenance** profoundly.
- Well written code can reduce the testing and maintenance effort. Because the testing and maintenance costs of software are much higher than the coding cost.
- the **goal of coding should be to reduce the testing and maintenance effort.**

Coding

- Hence, during coding the focus should be on developing programs that are **easy to read and understand**, and not simply on developing programs that are easy to write.
- **Simplicity and clarity** should be strived for during the coding phase.

Testing

- Testing is the major quality control measure used during software development.
- Its basic function is to ***detect defects*** in the software.
- During requirements analysis and design, the output is a document that is usually textual and non executable.
- After coding, computer programs are available that can be executed for testing purposes. This implies that testing not only has to uncover errors introduced during coding, but also errors introduced during the previous phases.
- Thus, the goal of testing is to uncover requirement, design, and coding errors in the programs.

Testing

- The starting point of testing is ***unit testing,*** where the different modules or components are tested individually.
- As modules are integrated in to the system, ***integration testing*** is performed, which focuses on testing the interconnection between modules.
- After the system is put together, ***system testing*** is performed. Here the system is tested against the system requirements to see if all the requirements are met and if the system performs as specified by the requirements.
- Finally, ***acceptance testing*** is performed to demonstrate to the client, on the real-life data of the client, the operation of the system.

Testing

- The testing process starts with ***a test plan*** that identifies all the testing-related activities that must be performed and specifies the schedule, allocates the resources, and specifies guidelines for testing.
- The test plan specifies conditions that should be tested, different units to be tested, and the manner in which the modules will be integrated.
- Then for different test units, a *test case specification document* is produced, which lists all the different test cases, together with the expected outputs.
- During the testing of the unit, the specified test cases are executed and the actual result compared with the expected output.
- The final output of the testing phase is the *test report* and the *error report,* or a set of such reports.
- Each test report contains the set of test cases and the result of executing the code with these test cases.
- The error report describes the errors encountered and the action taken to remove the errors.

Software Project Management Activities

- The main **goal** of software project management is to enable a group of software developer to work efficiently towards successful completion of project.
- Project Management activities can be classified into two categories:
  - Project Planning Activities
  - Project Monitoring & Control Activities

Project Planning Activities

- Involves several characteristics of the project and then planning the project activities based on the estimate made.
- Commence after the ***Feasibility study*** phase and before *Requirement Analysis*.
- Revised from time to time as project progresses.

Project Planning Activities

- Estimation
  - *Cost Estimation* – How much Cost?
  - *Duration Estimation* – How long to develop product?
  - *Effort Estimation*- How much Effort required?
- Scheduling
  - Schedule for manpower & other resources.
- Staffing
  - Staff Organization and staffing plans are made.
- Risk Management
  - Risk identification, analysis and abatement planning
- Miscellaneous Plans
  - Quality Assurance Plan, Configuration Management Plan, etc.

Project Monitoring & Control Activities

- Start once the development activities start.
- **Goal** is to ensure that the development proceeds as per plan.
- The plan changed whenever required to cope up with situation.

Software Quality

- Software must satisfy following factors
- Portability
  - Work on different hardware & OS
  - Easily interface with external devices
- Usability
  - Different categories of user can easily use the software
- Reusability
  - Different modules can be easily reused to develop new software
- Correctness
  - SRS must correctly implemented
- Maintainability
  - Errors can be easily corrected
  - New functionality added or modified easily

THANK YOU…

END OF UNIT-1