Software Engineering (CSE 415)

Introduction to Software Engineering

What is software?

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

Software Types

- Software products may be developed for a particular customer or may be developed for a general market.
- Two fundamental types of software products –

1) Generic: developed to be sold to a range of different customers who is able to buy (e.g. Software for PCs such as databases, word processors, drawings and project management tools).
2) custom: developed for a particular customer according to their specifications(e.g. Control system for electronic devices, air traffic control system)

- New software can be created by developing new programs, configuring generic software systems or reusing existing software.

Software's Dual Role

- Software is a ***product***
  - *Transforms* information - produces, manages, acquires, modifies, displays
- Software is a ***vehicle*** for delivering a product : Acts as the basis for :
  - Control of computer(operating system)
  - Communication of information (networking software)
  - Helps build and control other programs (software tools & environments)

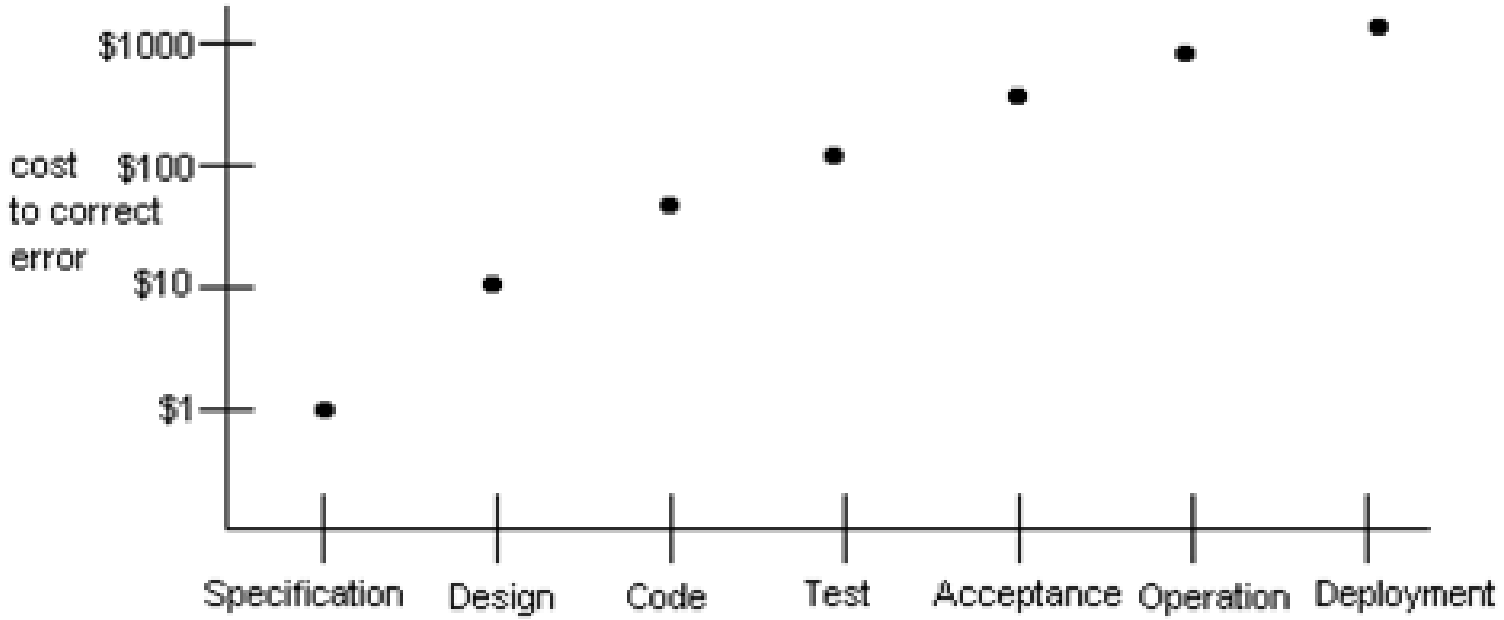
What is software engineering?

- Software engineering is an engineering discipline that is concerned with all aspects of software production.
- Software engineering is the application of systematic, disciplined, quantifiable approach to software development, operation and maintenance.
- Designing, building and maintaining large software systems in a cost-effective way.

Why is software engineering important?

- To develop methods for developing s/w that can scale up and be used to consistently develop high-quality s/w at low cost.
- To avoid costly errors caused by software.
  - Examples:
  - Lost Voyager Spacecraft (one bad line of code caused failure)
  - Commercial aircraft accidentally shot down during Gulf War (poor user interface)
- The earlier you detect errors, the lower the money you have to spend (i.e. save money a lot)
- Software engineers should adopt a systematic and organized approach to their work and use appropriate tools and techniques depending on the problem to be solved, the development constraints and the resources available.

Early Error Detection Saves Money



Hardware vs. Software

Hardware	Software
▪ Manufactured ▪ Wears out ▪ Built using components ▪ Relatively simple	▪ Developed/engineered ▪ Deteriorates ▪ Custom built ▪ Complex

Manufacturing vs. Development

- Once a hardware product has been manufactured, it is difficult or impossible to modify. 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.
- Unlike hardware, software costs are concentrated in design rather than production.

Software Characteristics

- Software is both a product and a vehicle for developing a product.
- Software is developed/engineered, not manufactured.
- Software does not wear out, but it does deteriorate.
- Most software is still custom-built.

Failure curves for Hardware

Hardware wears out over time

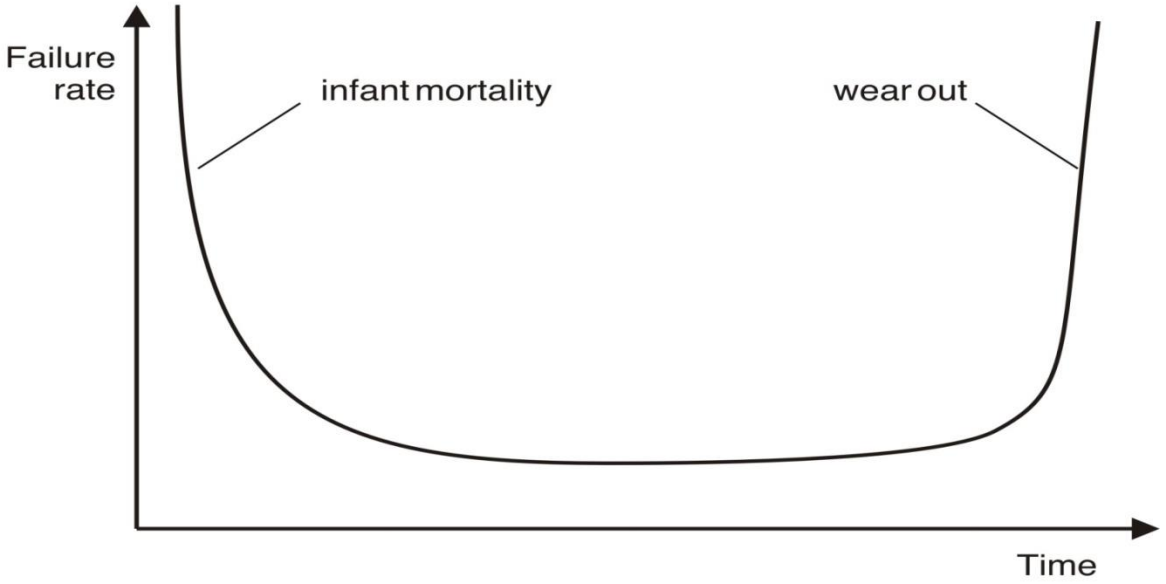


Figure 1 : Hardware Failure curve(the Bathtub curve)

Failure curves for Hardware

Hardware Failure(the Bathtub curve):

- Initially high
  - Design/Manufacturing defects
- Then failure rate drops
  - Defects corrected
- Again increases
  - Dust, vibration, abuse, high temperatures & Environmental maladies

Failure curves for Software

Software deteriorates over time

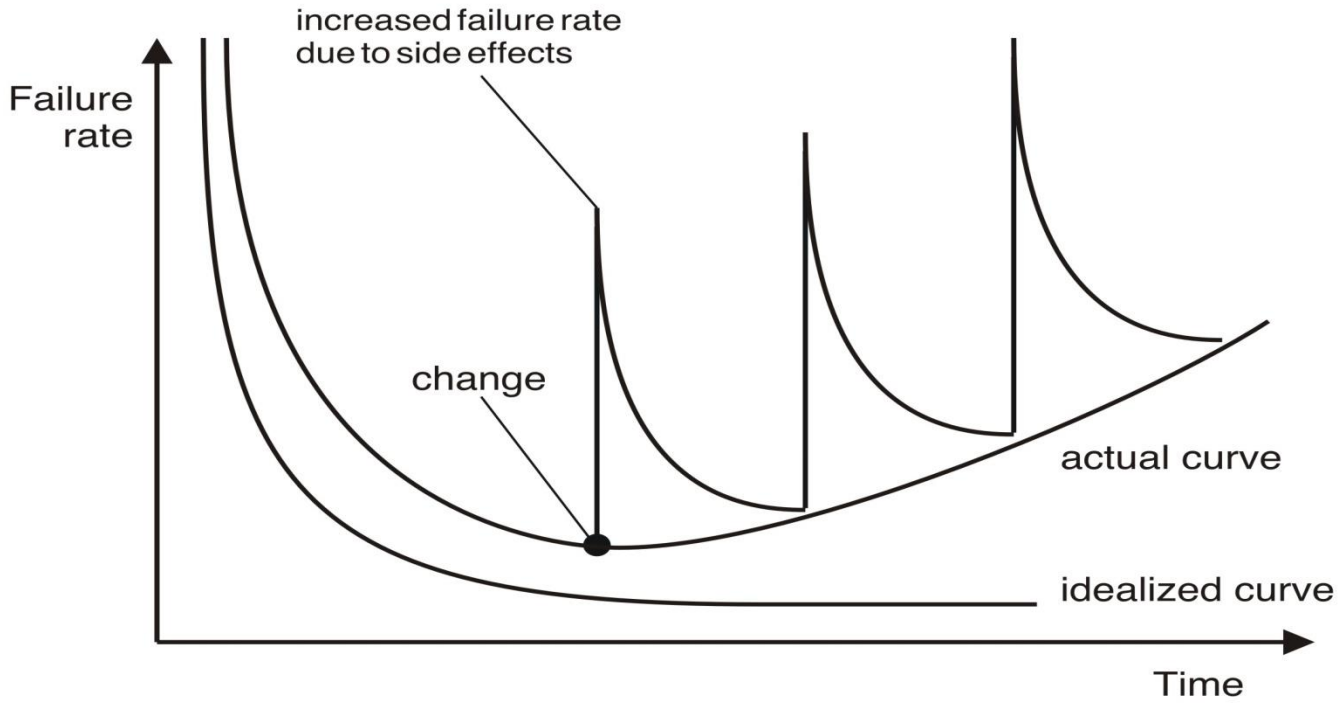


Figure 2 : Software Failure curve

Failure curves for Software

Software Failures:

- High failure rates early in life
  - Due to undiscovered defects
- Then failure rates drops & flattens (ideal curve)
  - Defects are corrected (without introduction of new bugs)
- Again failure rate increases (actual curve)
  - Due to **CHANGE** (most software will eventually undergo change)

Failure curves for Software

- How to reduce software deterioration?
  - Do better software design
  - Software Engineering methods strive to reduce the magnitude of the spikes & slope of the actual curve (Figure 2)

Legacy Software: *Why must it change?*

- It must be fixed to eliminate errors.
- It must be enhanced to implement new functional and non-functional requirements
- Software must be adapted to meet the needs of new computing environments or technology.
- Software must be enhanced to implement new business requirements.
- Software must be extended to make it interoperable with other more modern systems or databases.
- Software must be re-architected to make it viable within a network environment.

Software Myths

- Affect managers, customers (and other non-technical stakeholders) and practitioners
- Are believable because they often have elements of truth,

but …

- Invariably lead to bad decisions,

therefore …

- Insist on reality as you navigate your way through software engineering

Management Myths

- "We already have a book of standards and procedures for building software. It does provide my people with everything they need to know …"
- "If my project is behind the schedule, I always can add more programmers to it and catch up …"
  (a.k.a. "**The Mongolian Horde concept**")
- "If I decide to outsource the software project to a third party, I can just relax: Let them build it, and I will just pocket my profits …"

Customer Myths

- “A general statement of objectives is sufficient to begin writing programs - we can fill in the details later …”

- “Project requirements continually change but this change can easily be accommodated because software is flexible …”

Practitioner's Myths vs. Reality

- "Let's start coding ASAP, because once we write the program and get it to work, our job is done …"
  - It is only the beginning
- "Until I get the program running, I have no way of assessing its quality …"
  - Practice formal technical review
- Project requirements change continually and change is easy to accommodate in the software design.
  - Understand the requirements first , then write codes. It costs more to change later.

Practitioner's Myths vs. Reality

- "The only deliverable work product for a successful project is the working program …"
  - Working program is only one part of a software configuration that includes many elements

- "Software engineering is baloney. It makes us create tons of paperwork, only to slow us down …"
  - Software engineering in not about creating documents, it is about creating quality. Better quality leads to reduced rework and reduced rework results in faster delivery times

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.

What is CASE?

- Computer-Aided Software Engineering
- Software systems that are intended to provide automated support for software process activities.
- Covers a wide range of different types of programs that are used to support software process activities such as requirements analysis, system modelling, debugging and testing.
- CASE systems are often used for method support.
- Upper-CASE
  - Tools to support the early process activities of requirements and design
- Lower-CASE
  - Tools to support later activities such as programming, debugging and testing

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-Changing needs of customer
    - Dependability-Safety, Security, Reliability
    - Efficiency-Responsiveness, Processing time
    - Usability- User friendly, adequate documentation
    - Acceptability –Software must accepted by the users for which it was designed. This means it must be understandable, usable and compatible with other systems.

What are the key challenges in software development?

- High cost
- Difficult to deliver on time
- Low quality