Software Requirements and Specifications

Lecture 1

What is “Software” ?

❖Software =

It takes data, instruction

produce information.

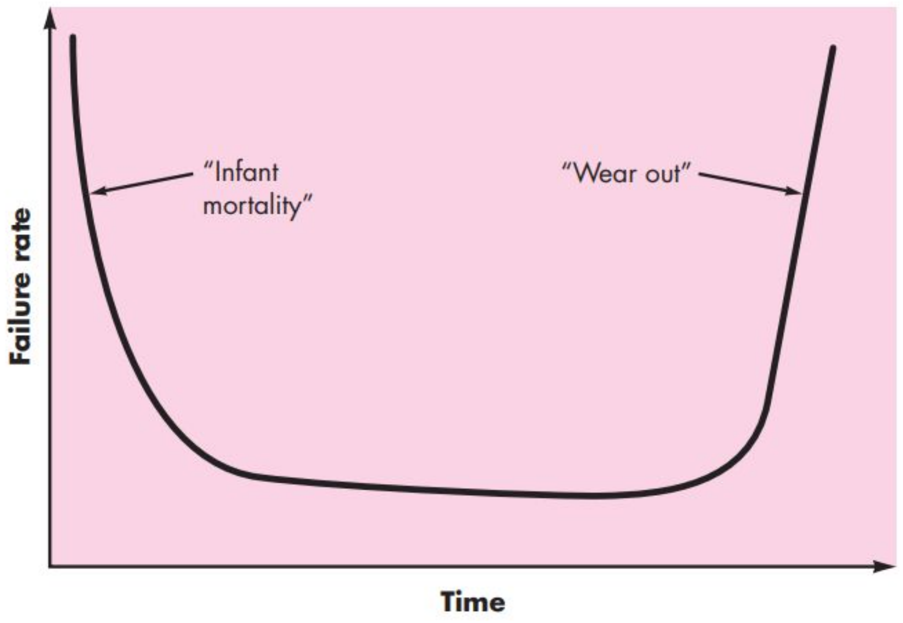
Software is:

- (1) instructions (computer programs) that when executed provide desired features, function, and performance;
- (2) data structures that enable the programs to adequately manipulate information, and
- (3) documentation that describes the operation and use of the programs

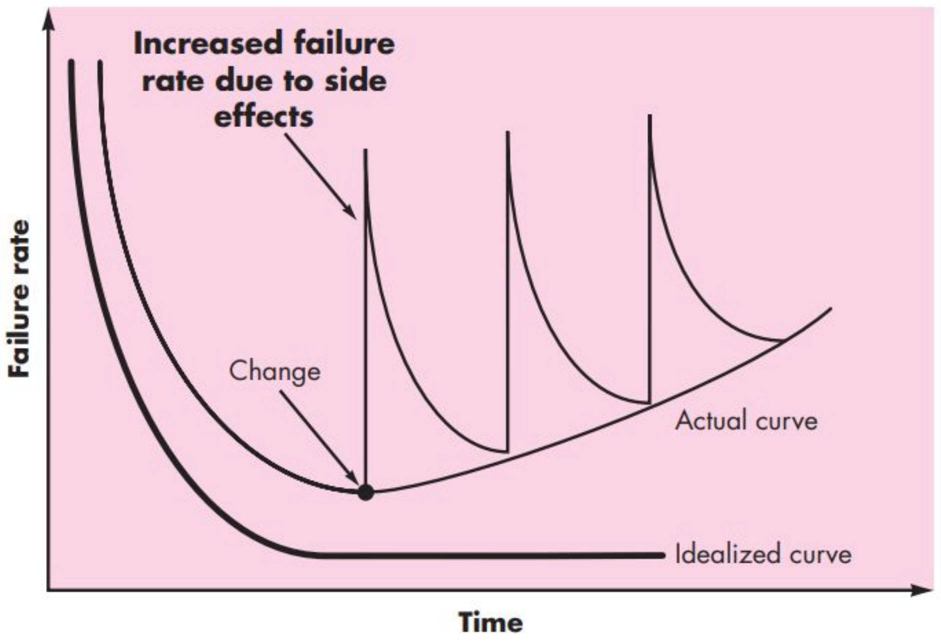
Software Applications

- system software [Ex: Operating system, drivers, networking software]
- application software [Ex: transaction processing, manufacturing process control]
- engineering/scientific software [Ex: Computer-aided design, system simulation]
- embedded software [Ex: key pad control for a microwave oven, fuel control, dashboard displays, and braking systems of automobile]
- Web applications [Ex: facebook, youtube]
- AI software [Ex: robotics, game playing]

Software doesn't wear out



Failure curve for hardware



Failure curve for software

Legacy Software

Why must it change?

- 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.

Characteristics of WebApps

- **Network intensiveness.**
- **Concurrency.**
- **Unpredictable load.**
- **Performance.**
- **Availability.**
- **Data driven.**
- **Content sensitive.**
- **Continuous evolution.**
- **Immediacy.** market quickly
- **Security.**
- **Aesthetics.** its look and feel.

A Process Framework

- Process = activities + action + tasks to create a product
- Activity = a broad objective (communication, design)
- Action = a set of tasks (architectural design, UI design, database design)
- Task = a small, but well-defined objective (unit testing)

Framework Activities

- Communication
- Planning
- Modeling
  - Analysis of requirements
  - Design
- Construction
  - Code generation
  - Testing
- Deployment

A Process Framework

Umbrella activities: are applicable across the entire software process

- Software project tracking and control
- Risk management
- Software quality assurance
- Formal technical reviews
- Measurement
- Software configuration management
- Reusability management
- Work product preparation and production

The Essence of Practice

Polya suggests:

1. *Understand the problem* (communication and analysis).
2. *Plan a solution* (modeling and software design).
3. *Carry out the plan* (code generation).
4. *Examine the result for accuracy* (testing and quality assurance).

Understand the Problem

- *Who has a stake in the solution to the problem?* That is, who are the stakeholders?
- *What are the unknowns?* What data, functions, and features are required to properly solve the problem?
- *Can the problem be compartmentalized?* Is it possible to represent smaller problems that may be easier to understand?
- *Can the problem be represented graphically?* Can an analysis model be created?

Plan the Solution

- *Have you seen similar problems before?* Are there patterns that are recognizable in a potential solution?
- *Has a similar problem been solved?* If so, are elements of the solution reusable?
- *Can subproblems be defined?* If so, are solutions readily apparent for the subproblems?
- *Can you represent a solution in a manner that leads to effective implementation?* Can a design model be created?

Carry Out the Plan

- *Does the solution conform to the plan?* Is source code traceable to the design model?

- *Is each component part of the solution provably correct?* Has the design and code been reviewed, or better, have correctness proofs been applied to algorithm?

Examine the Result

- *Is it possible to test each component part of the solution?* Has a reasonable testing strategy been implemented?

- *Does the solution produce results that conform to the data, functions, and features that are required?* Has the software been validated against all stakeholder requirements?

Hooker's General Principles

1. ***The Reason It All Exists***

   to provide value to its users.

2. ***KISS (Keep It Simple, Stupid!)***

   All design should be as simple as possible

3. ***Maintain the Vision***

   A clear vision is essential to the success of a software project

4. ***What You Produce, Others Will Consume***

   always specify, design, and implement knowing someone else will have to understand what you are doing

5. ***Be Open to the Future***

   Never design yourself into a corner. Always ask "what if,"

6. ***Plan Ahead for Reuse***

   Reuse saves time and effort. Use object-oriented rather than procedural

7. ***Think!***

   Placing clear, complete thought before action almost always produces better results

Software Myths

- Management myths:
  - **Myth**: 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?
  - **Reality**: is it used? Are software practitioners aware of its existence? Does it reflect modern software engineering practice? Is it complete? Is it adaptable?
  - **Myth**: If we get behind schedule, we can add more programmers and catch up (sometimes called the "Mongolian horde" concept).

- Customer myths:
  - **Myth**: A general statement of objectives is sufficient to begin writing programs—we can fill in the details later
  - **Reality:** Although a comprehensive and stable statement of requirements is not always possible, an ambiguous "statement of objectives" is a recipe for disaster.
  - **Myth**: Software requirements continually change, but change can be easily accommodated because software is flexible.
  - **Reality**: It is true that software requirements change, but the impact of change varies with the time at which it is introduced.

Software Myths

- Practitioner's myths:
  - **Myth**: Once we write the program and get it to work, our job is done.
  - **Reality**: someone once said that "the sooner you begin 'writing code,' the longer it'll take you to get done." 60 - 80 percent of all effort expanded after it is delivered to the customer for the first time
  - **Myth**: The only deliverable work product for a successful project is the working program.
  - **Reality**: A working program is only one part of a software configuration that includes many elements. A variety of work products (e.g., models, documents, plans) provide a foundation for successful engineering and, more important, guidance for software support.
  - **Myth**: Software engineering will make us create voluminous and unnecessary documentation and will invariably slow us down
  - **Reality**: Software engineering is not about creating documents. It is about creating a quality product.

How it all Starts

SafeHome:

- Every software project is precipitated by some business need—
  - the need to correct a defect in an existing application;
  - the need to the need to adapt a 'legacy system' to a changing business environment;
  - the need to extend the functions and features of an existing application, or
  - the need to create a new product, service, or system.