Software Engineering & Information System Design

Reference Book: Software Engineering by Roger S. Pressman

Chapter 1

Software & Software Engineering

What is Software?

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

(3) documentation that describes the operation and use of the programs.

What is Software? (Contd.)

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

  the term **manufacture** means**:** to make things, usually on a large scale, with tools and either physical labor or machinery

- Software doesn't "wear out".

- Although the industry is moving toward **component-based construction (reuse-based approach),** most software continues to be custom-built.

Hardware vs. Software

Hardware	Software
▪ Manufactured	▪ Developed/engineered
▪ Wears out (no longer usable)	▪ Deteriorates (progressively worse)
▪ Built using components	▪ Custom built
▪ Relatively simple	▪ Complex

Failure Curves for Hardware

- Also called the “bathtub curve”.
- At the beginning of the life of hardware it shows high failure rate as it contains many defects.
- By time, the manufacturers or the designers repair these defects and it becomes idealized or gets into the steady state and continues.
- But after that, as time passes, the failure rate rises again and this may be caused by excessive temperature, dust, vibration, improper use and so on and at one time it becomes totally unusable. This state is the “wear out” state.

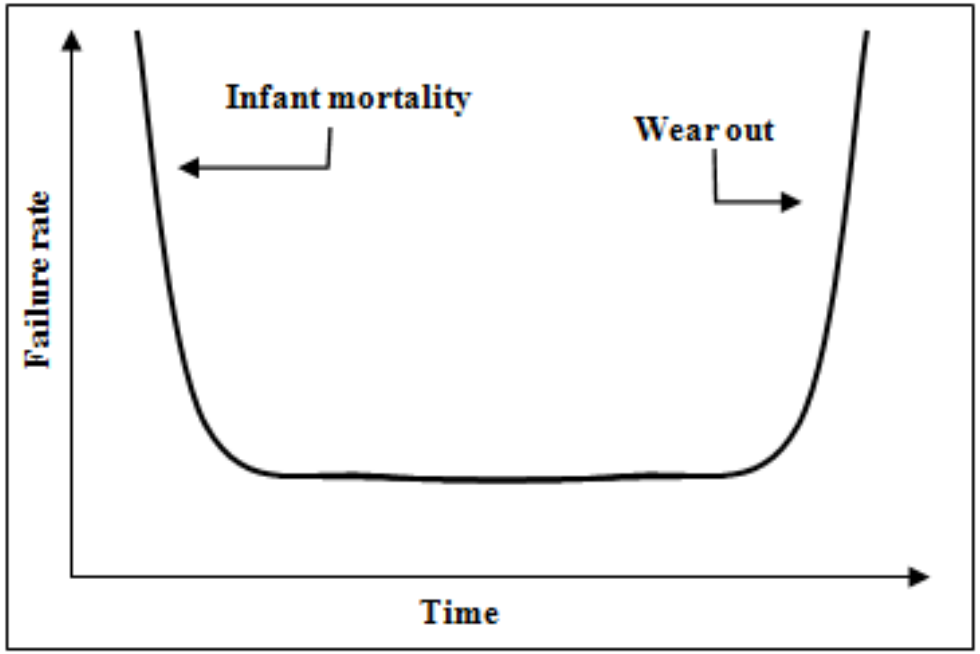
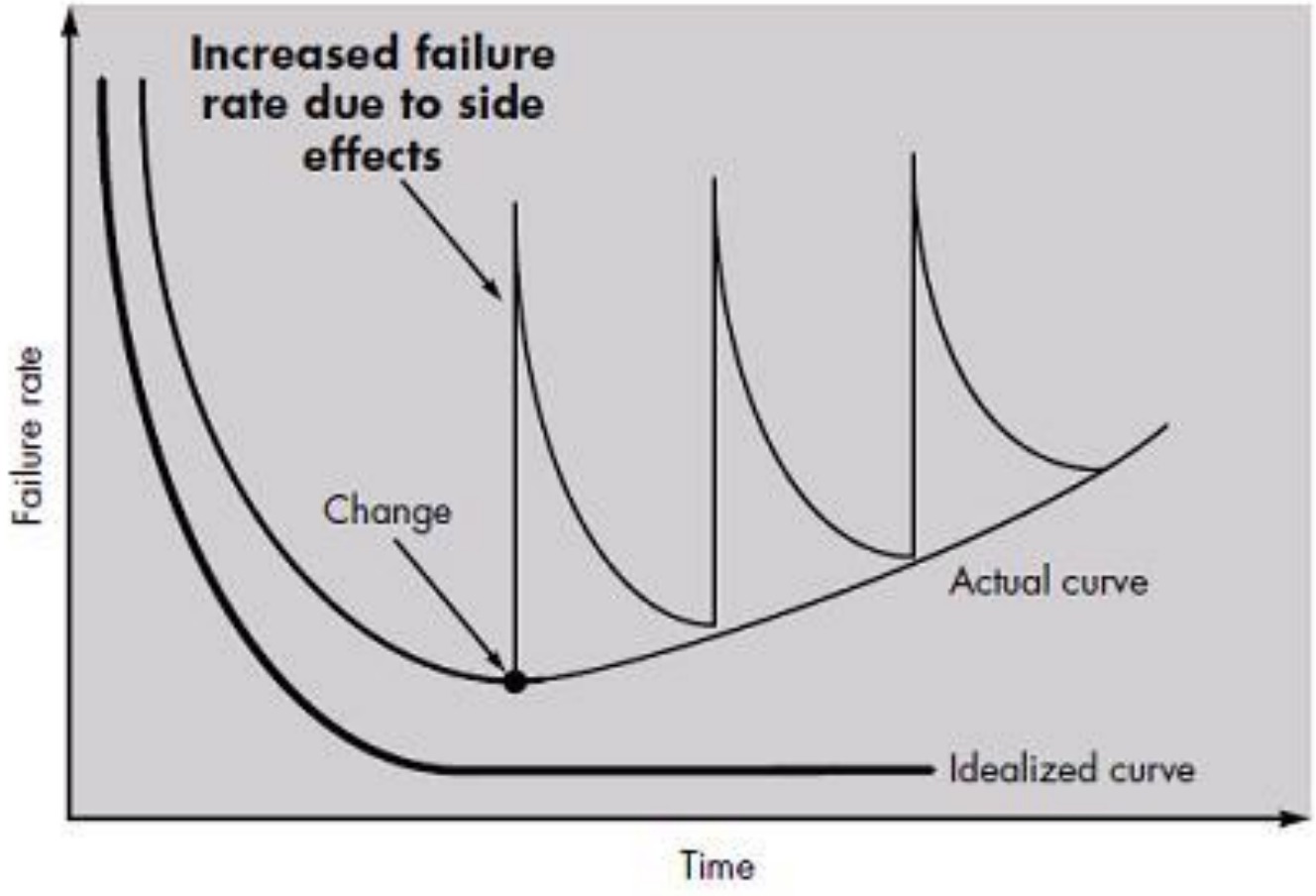


Figure 01: Failure curve for hardware

Failure Curves for Software



Software Applications

- System software: such as compilers, editors, file management utilities
- Application software: stand-alone programs for specific needs
- Engineering/scientific software: such as automotive stress analysis, molecular biology, orbital dynamics etc
- Embedded software: resides within a product or system. (key pad control of a microwave oven, digital function of dashboard display in a car)

Software Applications (Contd.)

- Product-line software: focus on a limited marketplace to address mass consumer market. (word processing, graphics, database management)

- WebApps (Web applications): network centric software. As web 2.0 emerges, more sophisticated computing environments is supported integrated with remote database and business applications.

- AI software: uses non-numerical algorithm to solve complex problem. Robotics, expert system, pattern recognition, game playing

Legacy Software

- Informally characterized as *old software* *that is still performing a useful job for the community*
- Example: Consider a software developed years ago using early versions of Fortran or other languages

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.

Software Engineering

IEEE defines software engineering as:

The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software

Fritz Bauer, a German computer scientist, defines software engineering as:

Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and work efficiently on real machines.

Software engineering is important because it enables us to build complex systems in a timely manner and with high quality

Software Process

- Software process is a collection of:
  - Activities, actions, and tasks
- Performed when work product is to be created
- Generic software engineering encompasses 5 activities:
  - Communication
  - Planning
  - Modeling
  - Construction
  - Deployment

Software Engineering Practice

- Understand the problem (communication and analysis)
- Plan a solution (modeling a software design)
- Carry out the plan (code generation)
- Examine the result for accuracy (testing and quality assurance)

Umbrella Activities in Software Engineering

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

Practice Problems

- Why it is important to understand the customer's problem before creating a software solution?
- Often the practitioners of software engineering believe that "Once we write the program and get it to work, our job is done".

  - What's the reality?

- There is a myth among the customer that "Software requirements continually change, but change can be easily accommodated because software is flexible".

  - What's the reality?