Chapter 1
Software and Software Engineering

Software Engineering: A Practitioner's Approach, 6th edition
by Roger S. Pressman

Software’s Dual Role

- Software is a product
  - *Transforms* information - produces, manages, acquires, modifies, displays, or transmits information;
  - Delivers computing potential of hardware and networks
- Software is a vehicle for delivering a product
  - Controls other programs (operating system);
  - Effects(实现) communications (networking software);
  - Helps build other software (software tools & environments)

Software Applications

- system software
- application software
- engineering/scientific software[CAD etc.]
- embedded software
- product-line software
- web applications
- AI software

Hardware vs. Software

Hardware

- Manufactured
- Wears out[磨损]
- Built using components
- Relatively simple

Software

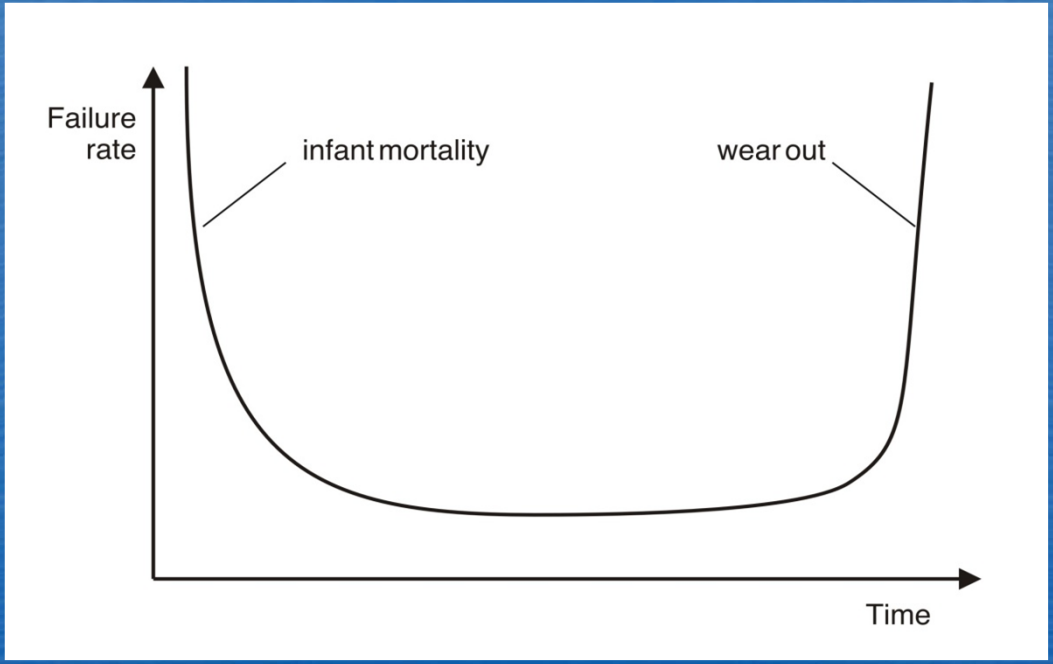
- Developed/engineered
- Deteriorates[恶化]
- Custom built[根据需求定制的]
- Complex

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.

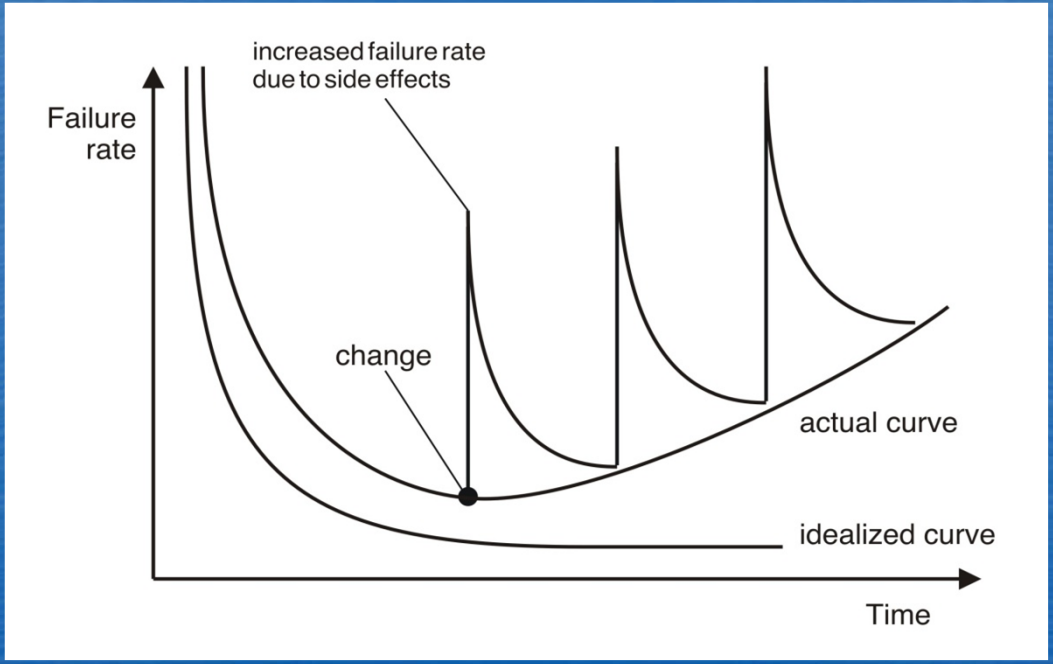
Wear out vs. Deterioration

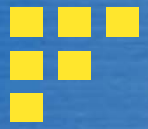
Hardware wears out over time



Wear vs. Deterioration

Software deteriorates over time





Component Based vs. Custom Built

- Hardware products typically employ many standardized design components.
- Most software continues to be custom built.
- The software industry does seem to be moving (slowly) toward component-based construction.

Software *Complexity*

I believe the hard part of building software to be **the specification, design, and testing of this conceptual construct**, not the labor of representing it and testing the fidelity of the representation.

If this is true, building software will always be hard. There is inherently no silver bullet.

- Fred Brooks, “No Silver Bullet”

http://www.computer.org/computer/homepage/misc/Brooks/

No Silver Bullet： 没有仙丹 只没有一下子就能解决问题的方法

Legacy Software

Why must it change?

Changeablility

- 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.

E-Type Systems

- **E-Type Systems:**
  Software that has been implemented in a real-world computing context and will therefore evolve over time

Software Evolution[Lehman定律]

- The Law of Continuing Change [持续变化规律] (1974): E-type systems must be continually adapted else they become progressively less satisfactory.
- The Law of Increasing Complexity [复杂性增长规律] (1974): As an E-type system evolves its complexity increases unless work is done to maintain or reduce it.
- The Law of Self Regulation [自我调控规律] (1974): The E-type system evolution process is self-regulating with distribution of product and process measures close to normal.

Software Evolution…

- The Law of Conservation of Organizational Stability [组织稳定性守恒规律] (1980): The average effective global activity rate in an evolving E-type system is invariant over product lifetime.
- The Law of Conservation of Familiarity [保证通晓性规律] (1980): As an E-type system evolves all associated with it, developers, sales personnel, users, for example, must maintain mastery of its content and behavior to achieve satisfactory evolution.
- The Law of Continuing Growth [持续增长规律] (1980): The functional content of E-type systems must be continually increased to maintain user satisfaction over their lifetime.

Software Evolution…

- The Law of Declining Quality [质量衰减规律] (1996): The quality of E-type systems will appear to be declining unless they are rigorously maintained and adapted to operational environment changes.
- The Feedback System Law [反馈系统规律] (1996): E-type evolution processes constitute multi-level, multi-loop, multi-agent feedback systems and must be treated as such to achieve significant improvement over any reasonable base.

Source: Lehman, M., et al, "Metrics and Laws of Software Evolution—The Nineties View," Proceedings of the 4th International Software Metrics Symposium (METRICS '97), IEEE, 1997, can be downloaded from http://www.ece.utexas.edu/~perry/work/papers/feast1.pdf

Software Myths[谬论]

- 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

Software Myths

- If we get behind schedule, we can add more programmers and catch up.
- A general statement about objectives is sufficient to begin building programs.
- Change in project requirements can be easily accommodated because software is flexible.

Software Myths

- Once we write a working program, we're done.
- 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 too much documentation and will slow us down.

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--蒙古部落的概念**")
- "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

- "Let's start coding ASAP, because once we write the program and get it to work, our job is done …"
- "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 is baloney[胡扯]. It makes us create tons of paperwork, only to slow us down …"