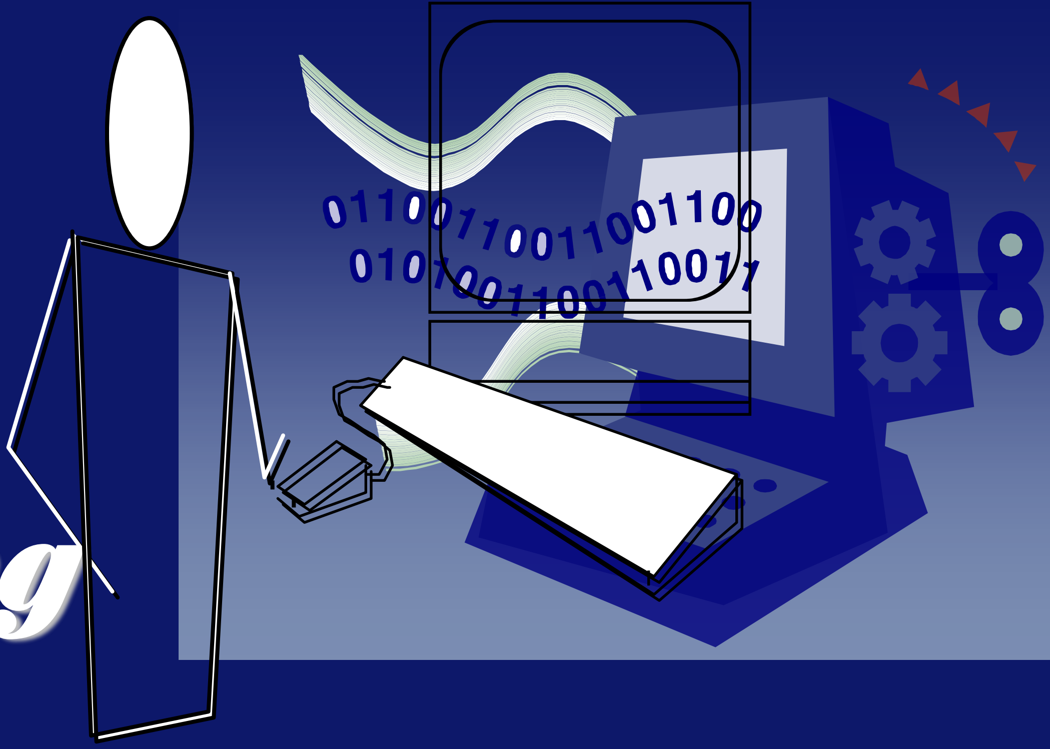


Software Engineering

Chapter 1

Software and Software Engineering

Moon kun Lee
Division of Electronics & Information Engineering
Chonbuk National University

Software's Dual Role

- Software is a product
  - Delivers computing potential
  - Produces, manages, acquires, modifies, displays, or transmits information
- Software is a vehicle for delivering a product
  - Supports or directly provides system functionality
  - Controls other programs (e.g., an operating system)
  - Effects communications (e.g., networking software)
  - Helps build other software (e.g., software tools)

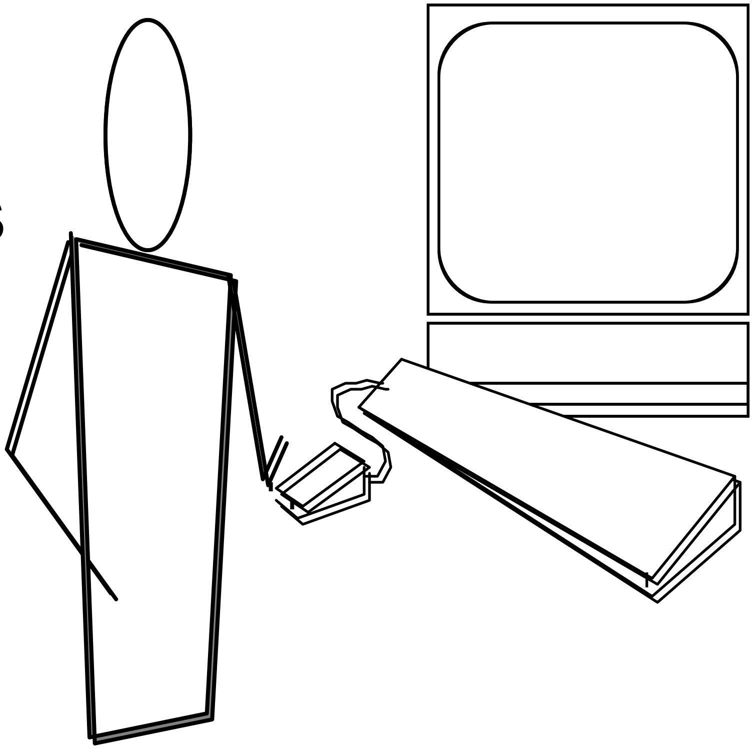
Evolution of Software

- The early year (~mid 60s)
  - Batch orientation
  - Limited distribution
  - Custom software
- The second era (~early 70s)
  - Multiuser
  - Real-time
  - Database
  - Product software
- The third era (~late 80s)
  - Distributed systems
  - Embedded "intelligence"
  - Low cost hardware
  - Consumer impact
- The fourth era (~2000)
  - Powerful desk-top systems
  - Object-oriented technology
  - Expert systems
  - Artificial neural networks
  - Parallel computing
  - Network computers
- The Fifth Era (~2010)
  - Web-based
  - Ubiquitous: office + car + home
  - RFID
  - LBS(Location-Based Service)
- The Sixth Era (2010~2020)
  - Smart Service
  - Cloud Service

What is Software?

Software is a set of items or objects that form a "configuration" that includes

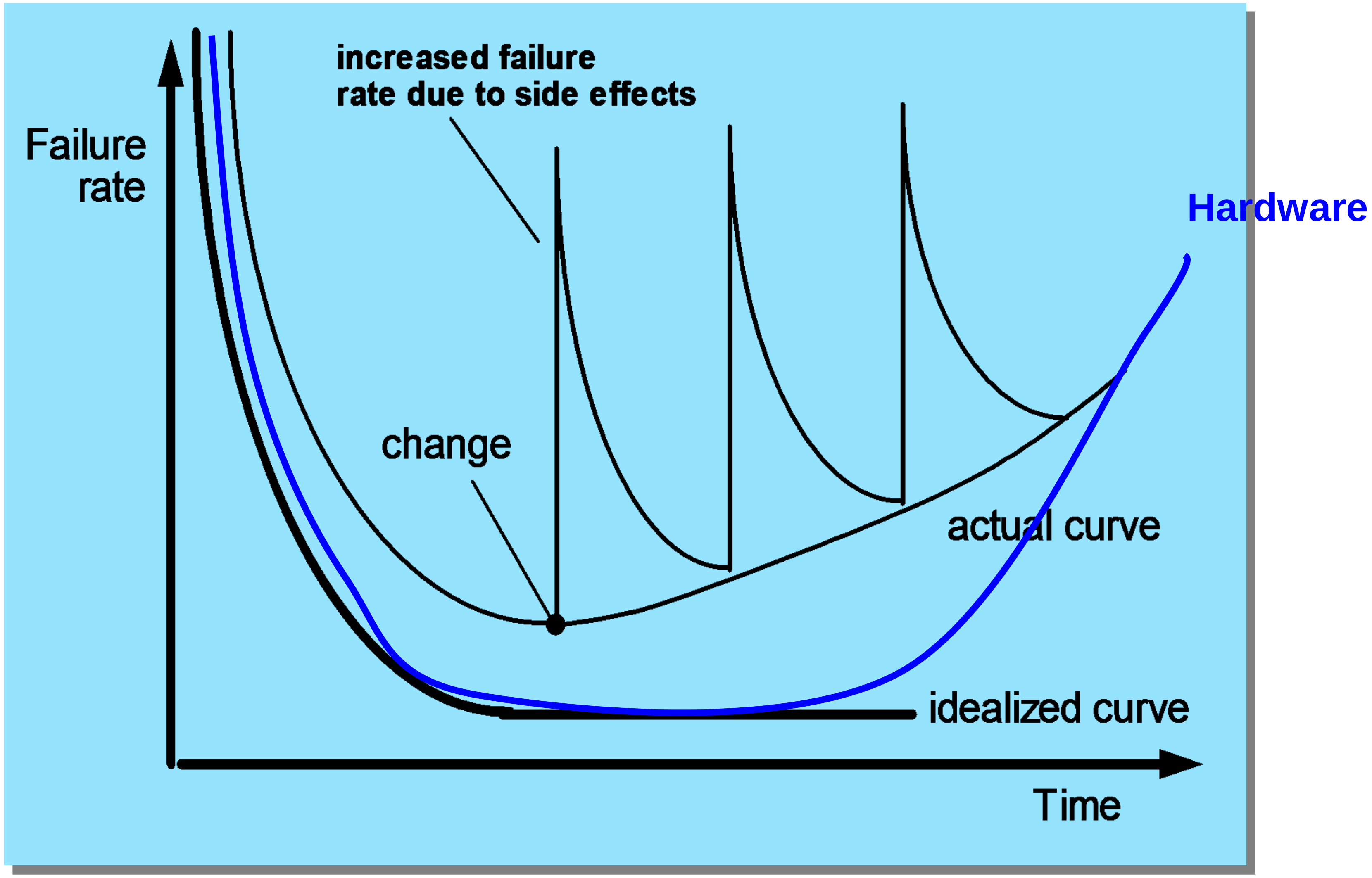
- programs
- documents
- data ...



What is Software?

- software is engineered
- software doesn’t wear out
- software is complex

Wear vs. Deterioration



Software Applications

- system software
  - Def: a collection of programs written to service other programs
  - Determinate
  - Indeterminate
- application software
- engineering/scientific software
- embedded software
- product-line software
- WebApps (Web applications)
- AI software

Software—New Categories

- Ubiquitous computing—wireless networks
- Netsourcing—the Web as a computing engine
- Open source—"free" source code open to the computing community (a blessing, but also a potential curse!)
- Also … (see Chapter 32)
  - Data mining
  - Grid computing
  - Cognitive machines
  - Software for nanotechnologies

✶ *Grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed "autonomous" resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements.*

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.

Software Evolution

- The Law of Continuing Change (1974): E-type systems must be continually adapted else they become progressively less satisfactory.
- The Law of Increasing Complexity (1974): As an E-type system evolves its complexity increases unless work is done to maintain or reduce it.
- The Law of Self Regulation (1974): The E-type system evolution process is self-regulating with distribution of product and process measures close to normal.
- The Law of Conservation of Organizational Stability (1980): The average effective global activity rate in an evolving E-type system is invariant over product lifetime.
- The Law of Conservation of Familiarity (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 (1980): The functional content of E-type systems must be continually increased to maintain user satisfaction over their lifetime.
- The Law of Declining Quality (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 (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**

E-type systems: software that has been implemented in a real-world computing context and will therefore evolve over time.

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

Examples of the Myths

- Management:
  - We already have a book that is full of standards and procedures for building software. Won't that provide my people with everything they need to know?
  - If we get behind schedule, we can add more programmers and catch up.
  - If I decide to outsource the software project to a third party, I can just relax and let that form build it.
- Customer:
  - A general statement of objectives is sufficient to begin writing programs- we can fill in the details later.
  - Project requirements continuously change, but change can be easily accommodated because software is flexible.
- Practitioners:
  - 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 will make us create voluminous and unnecessary documentation and will invariably slow us down.