Chapter 1

Software and Software Engineering

Scope of Software Engineering

- Historical Aspects
  - 1968 NATO Conference, Garmisch
  - Aim: to solve the “Software Crisis”
  - Software is delivered
    - ***Late***
    - ***Over budget***
    - ***Unreliable*** (with residual faults)

Scope of Software Engineering
(cont.)

- always with complaints like
  - why does it ***take so long*** to get software finished?
  - why are development ***costs so high***?
  - why ***can't we find all the errors*** before we give to customers?
  - why do we continue to have ***difficulty in measuring progress*** as software is being developed?

Real Cases

- *84% of software projects were not completed on time and within budget* (a survey conducted by Standish Group)
  - 8000 projects in US in 1995
  - more than 30% were cancelled
  - 189% over budget

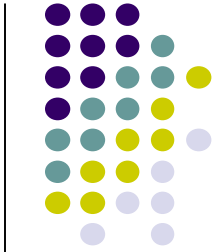
Real Cases (cont.)

- Bank of America Master Net System
  - Trust business, 1982
  - Spend 18 months in deep research & analysis of the target system
    - Original budget: 20 million
    - Original schedule: 9 month, due to 1984/12/31
    - Not until March 1987, and spent 60 million
    - Lost 600 millions business
  - Eventually, gave up the software system and 34 billion trust accounts transferred

Real Cases (cont.)

- 百億學費！新系統大崩壞　南山人壽600萬保戶成白老鼠 (ETtoday新聞雲，2019/9/19)
  - 自2014年啟動的全系統導入計畫，將50多年累積的近百套複雜的舊系統，整合為單一平台
  - 由國際軟體大廠思愛普（SAP）負責，預計三年完成
  - 從原先47億經過逐年追加預算，至2019年總經費已高達101億元
  - 除了一次想要將原本近百套複雜的舊系統，整合為單一平台太複雜外，最關鍵的或許是**南山新團隊對舊系統不熟悉**
  - 另外，**在系統尚未穩定（系統太敏銳也是尚未穩定），人員不熟練時即停掉舊系統，啟用新系統**

Software's Dual Role

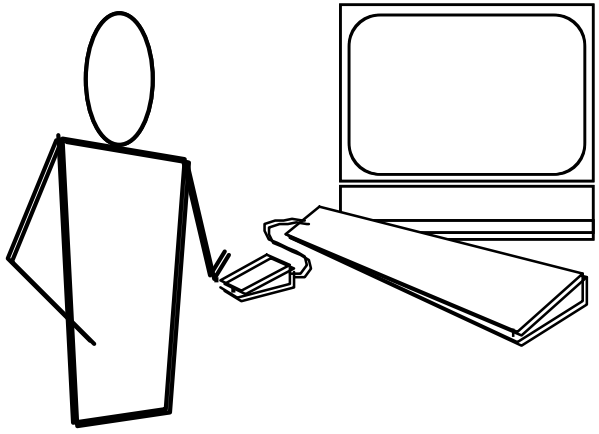


- Software is a product
  - Delivers computing potential
  - Produces, manages, acquires, modifies, displays, or transmits information
- Software is a vehicle for delivering a product
  - Supports or directly provides system functionality
  - Controls other programs (e.g., an operating system)
  - Effects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

What is Software?

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

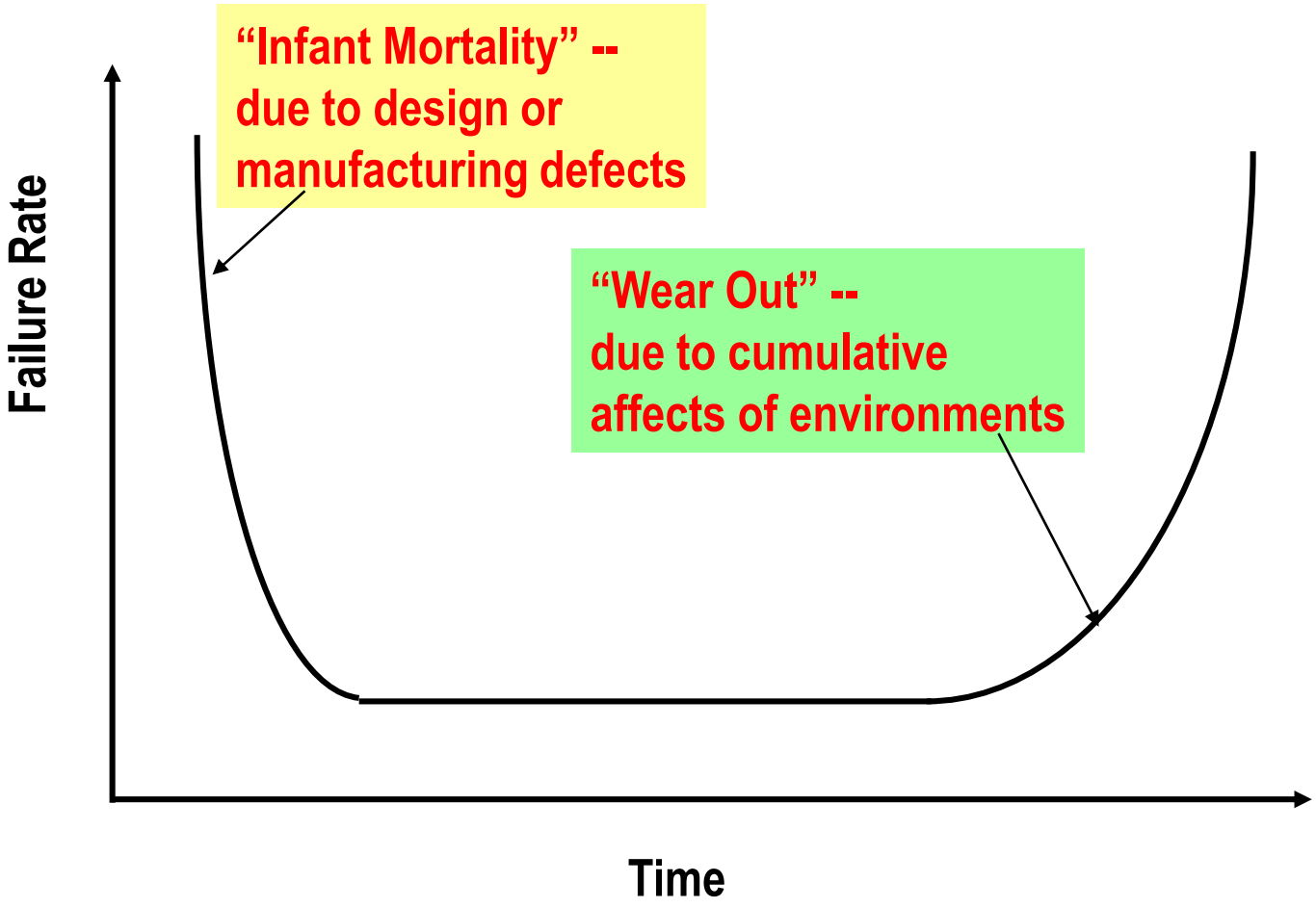
- **instructions** (programs)
- **data structures**
- **documents**



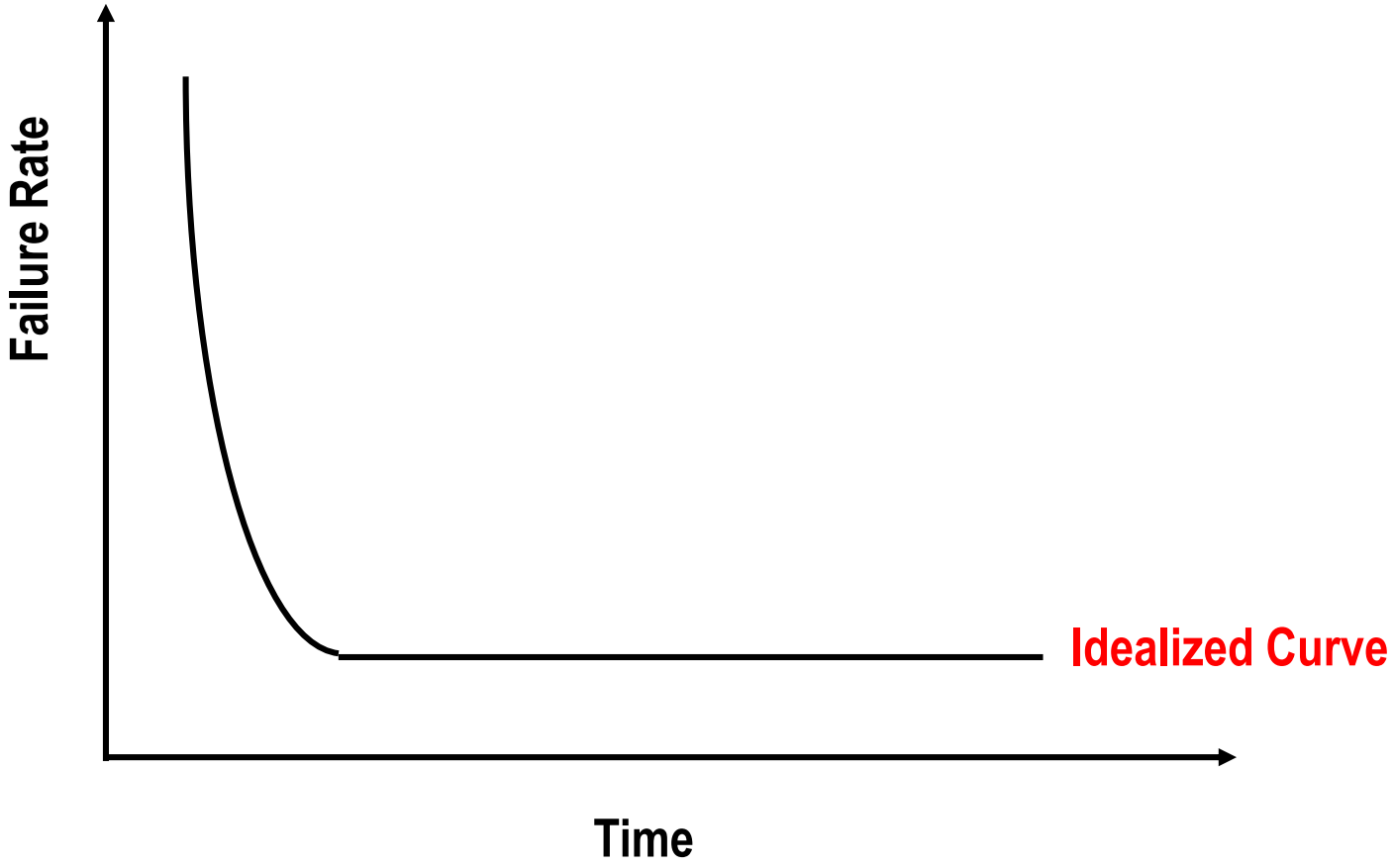
Software Characteristics

- Software is ***developed/engineered***, not ***manufactured***
- Software ***doesn't wear out*** but "deteriorating"; no spare parts
- Software is complex
  - Most software continuing to be custom built

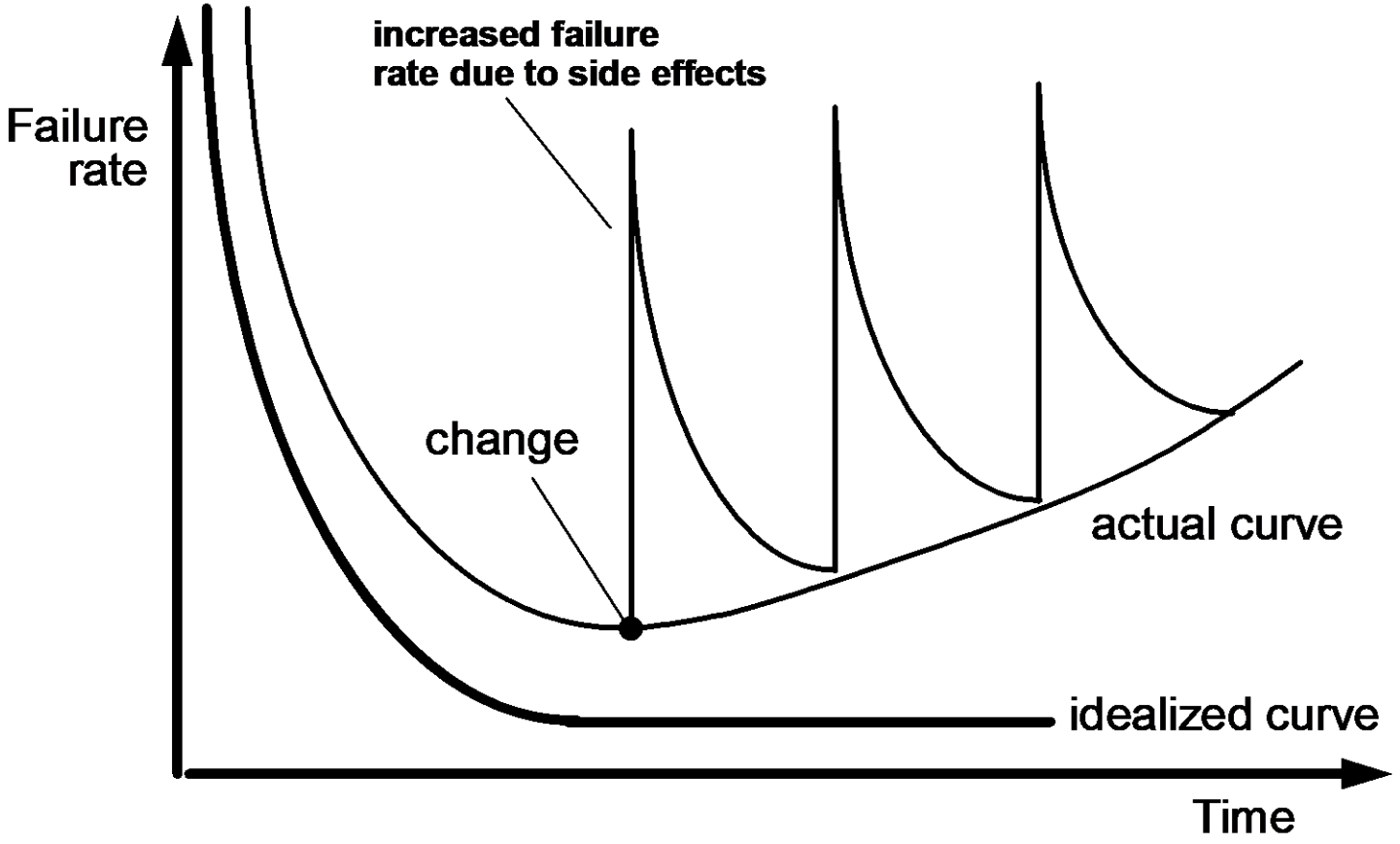
Product Bathtub Curve Model



Software Idealized Curve



Software Actual Failure Curve

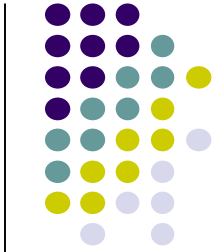


Software Application Domains

Seven broad categories

- System software
- Application software
- Engineering/scientific software
- Embedded software
- Product-line software
- Web/mobile software
- Artificial intelligence software

Legacy Software

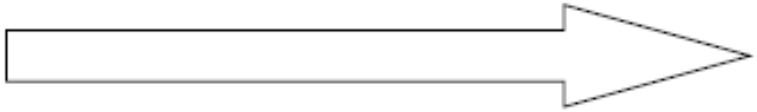


What is legacy software?
Why must it change?

- 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.

What is Software Engineering?

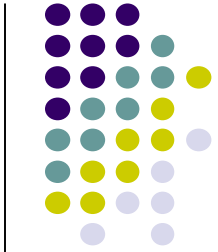
Software Engineering



Real World

Software World

What is Software Engineering? (cont.)



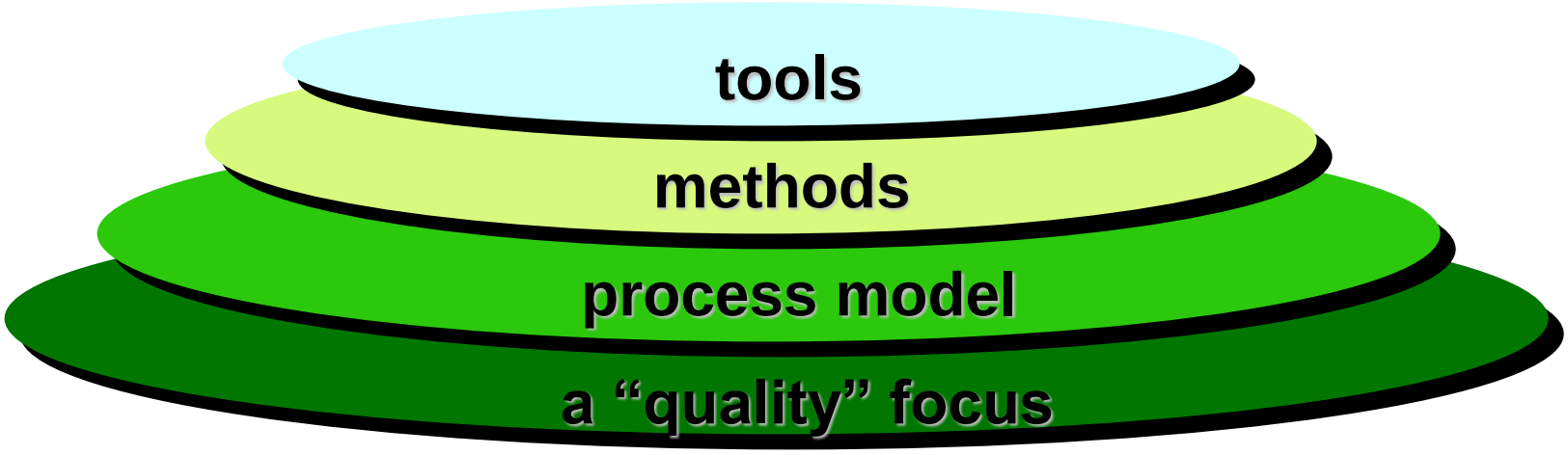
- The seminal definition:

  The establishment and use **of sound engineering principles** in order to obtain **economically** software that is **reliable** and works efficiently on real machines.

What is Software Engineering? (cont.)

- The IEEE definition:
  - (1) The application of a **systematic**, **disciplined**, **quantifiable** approach to the development, operation, and maintenance of software; that is, the application of engineering to software.
  - (2) The study of approaches as in (1).

Software Engineering is a Layered Technology



A Process Framework

Process framework

Framework activities

- **work tasks**
- **work products**
- **milestones & deliverables**
- **QA checkpoints**

Umbrella Activities

Framework Activities

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

Umbrella Activities

- Software project management
- Formal technical reviews
- Software quality assurance
- Software configuration management
- Document preparation and production
- Reusability management
- Measurement
- Risk management

The Essence of Practice

- George Polya, in a book written in 1945 (!), describes the essence of software engineering practice …
  - ***Understand the problem*** (communication and analysis).
  - ***Plan a solution*** (modeling and software design).
  - ***Carry out the plan*** (code generation).
  - ***Examine the result for accuracy*** (testing and quality assurance).
- At its core, good practice is *common-sense problem solving*

Core Software Engineering Principles

- David Hooker, 1996
- focus on software engineering as a whole
  - *Provide value to the customer and the user*
  - *KIS—keep it simple!*
  - *Maintain the product and project "vision"*
  - *What you produce, others will consume*
  - *Be open to the future*
  - *Plan ahead for reuse*
  - *Think!*

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

Software Myths (Management)

- Myth1: We already have a book that's full of standards and procedures for building s/w, won't that provide my people with everything they need to know?
- Myth2: If we get behind schedule, we can add more programmers and catch up.
- Myth3: If I decide to outsource the software project to a third party, I can just relax and let that firm build it.

Software Myths (Customer)

- Myth1*:* A general statement of objectives is sufficient to begin writing programs – we can fill in the details later.
- Myth2*:* Project requirements continually change, but change can be easily accommodated because software is flexible.

Software Myths (Practitioner)

- Myth1: Once we write the program and get it to work, our job is done.
  - *Fact:* the sooner you begin writing code, the longer it will take you to get done.
- Myth2: Until I get the program "running," I have no way of assessing its quality.
- Myth3: The only deliverable work product for a successful project is the working program.
- Myth4: Software engineering will make us create voluminous and unnecessary documentation and will invariable slow us down.

Conclusions

- Software has become the key element in the evolution of computer-based systems and products
- Software has evolved into a industry in itself
  - Yet still have trouble developing high-quality software on time and within budget
- The intent of software engineering is to provide a **framework** for building **high-quality** software