COURSE NAME

SOFTWARE ENGINEERING

CSC 3114

(UNDERGRADUATE)

CHAPTER 1

SOFTWARE & SOFTWARE ENGINEERING

PROF. DR. KAMRUDDIN NUR
PROFESSOR, CS, AIUB
https://cs.aiub.edu/profile/kamruddin

WHY SYSTEM FAILS?

- The system fails to meet the business requirements for which it was developed. The system is either abandoned or expensive adaptive maintenance is undertaken.
- There are performance shortcomings in the system, which make it inadequate for the users' needs. Again, it is either abandoned or amended incurring extra costs.
- Errors appear in the developed system causing unexpected problems. Patches have to be applied at extra cost.
- Users reject the implemented system, lack of involvement in its development or lack of commitment to it.
- Systems are initially accepted but over time become un-maintainable and so pass into disuse.

SCOPE OF SOFTWARE ENGINEERING

- The aim of Software Engineering is to solve Software Crisis:
  - Late
  - Over budget
  - Low quality with lots of faults
- Software crisis is still present over 35 years later!

SOFTWARE CHARACTERISTICS

- A logical (intangible) rather than a physical system element
- Being developed or engineered, but not being manufactured
- Software cost concentrating in engineering, not in materials
- Software does not "wearing out" but "deteriorating"(not destroyed after lifetime like hardware, but backdated by aging that needs to update)
- Software is a 'differentiator' (different sub-systems, e.g. cashier's workstation in a supermarket)
- Without "spare parts" in software maintenance (no extra useless features in software)
- Most software continuing to be custom built (based on the requirements)

GOAL: COMPUTER SCIENCE VS. SOFTWARE ENGINEERING

- CS: to investigate a variety of ways to produce S/W, some good and some bad
- SE: to be interested in only those techniques that make sound economic sense

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

- A structured set of activities required to develop a software system
- The way we produce software, including:

1. Requirements Analysis
2. Designing/Modeling
3. Coding /Development
4. Testing
5. Implementation/Integration phase
6. Operation/Maintenance
7. Documentation

GOOD & BAD SOFTWARE

- Good software is maintained—bad software is discarded

- Different types of maintenance
  - Corrective maintenance [about 20%]
    - Modification to fix a problem
  - Enhancement [about 80%]
    - Perfective maintenance (modification to improve usability,…) [about 60%]
    - Adaptive maintenance (modification to keep up-to-date) [about 20%]
    - Preventive maintenance (modification to avoid any future error) [about 20%]

FAULTS IN SOFTWARE DEVELOPMENT PHASES

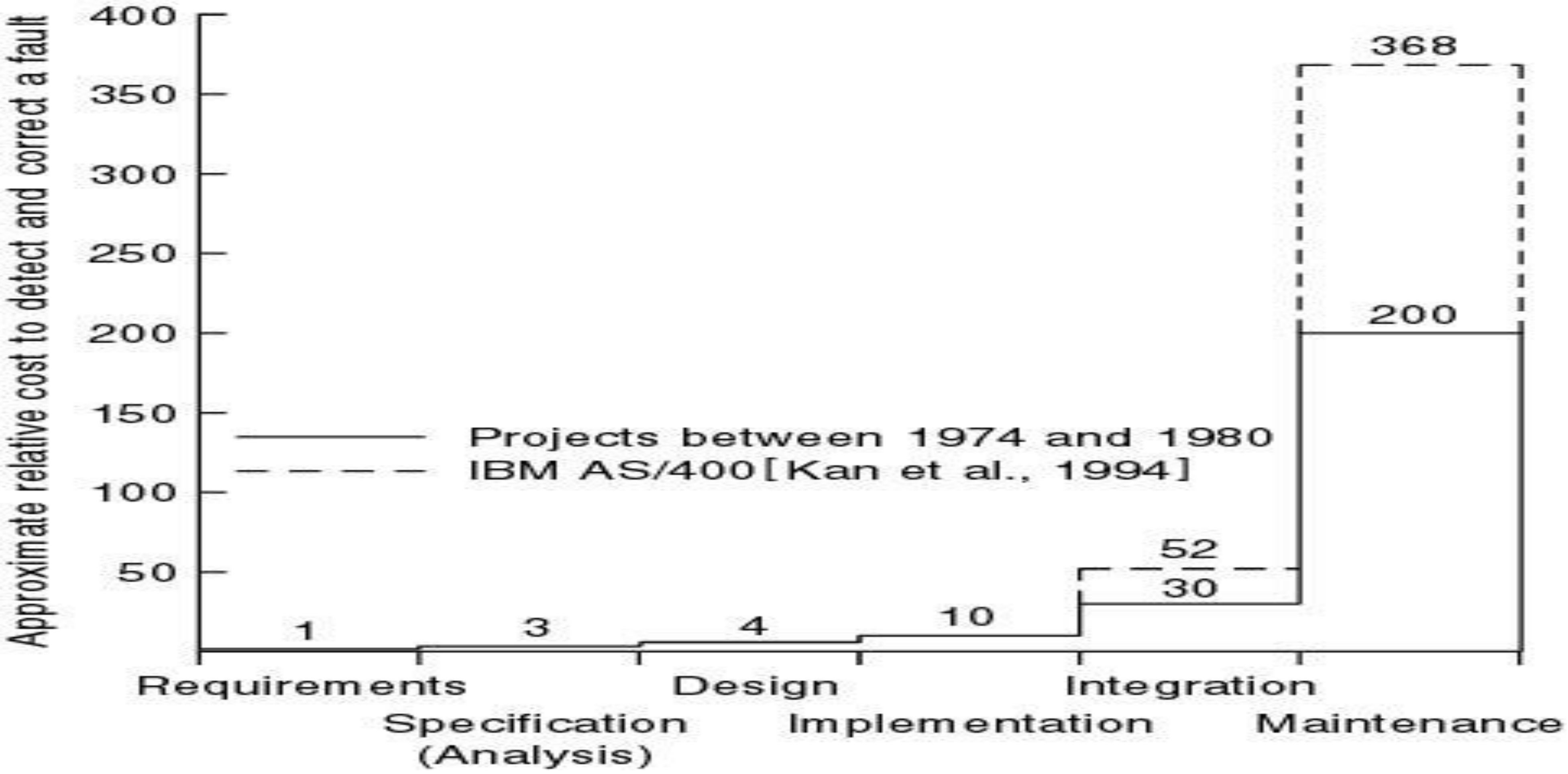
- 60 to 70 percent of faults are specification and design faults

- Data of Kelly, Sherif, and Hops [1992]
  - 1.9 faults per page of specification
  - 0.9 faults per page of design
  - 0.3 faults per page of code

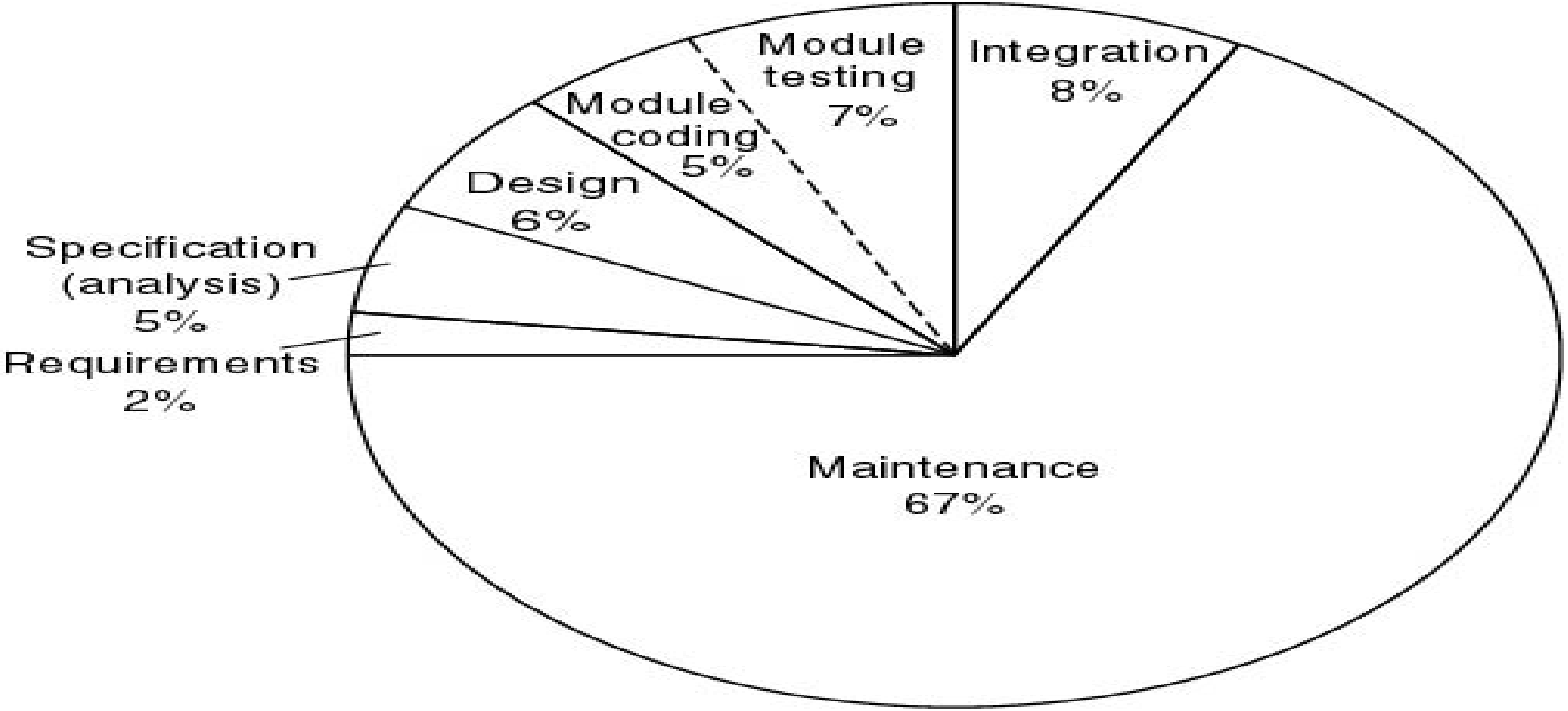
- Data of Bhandari et al. [1994]

  Faults at end of the design phase of the new version of the product
  - 13% of faults from previous version of product
  - 16% of faults in new specifications
  - 71% of faults in new design

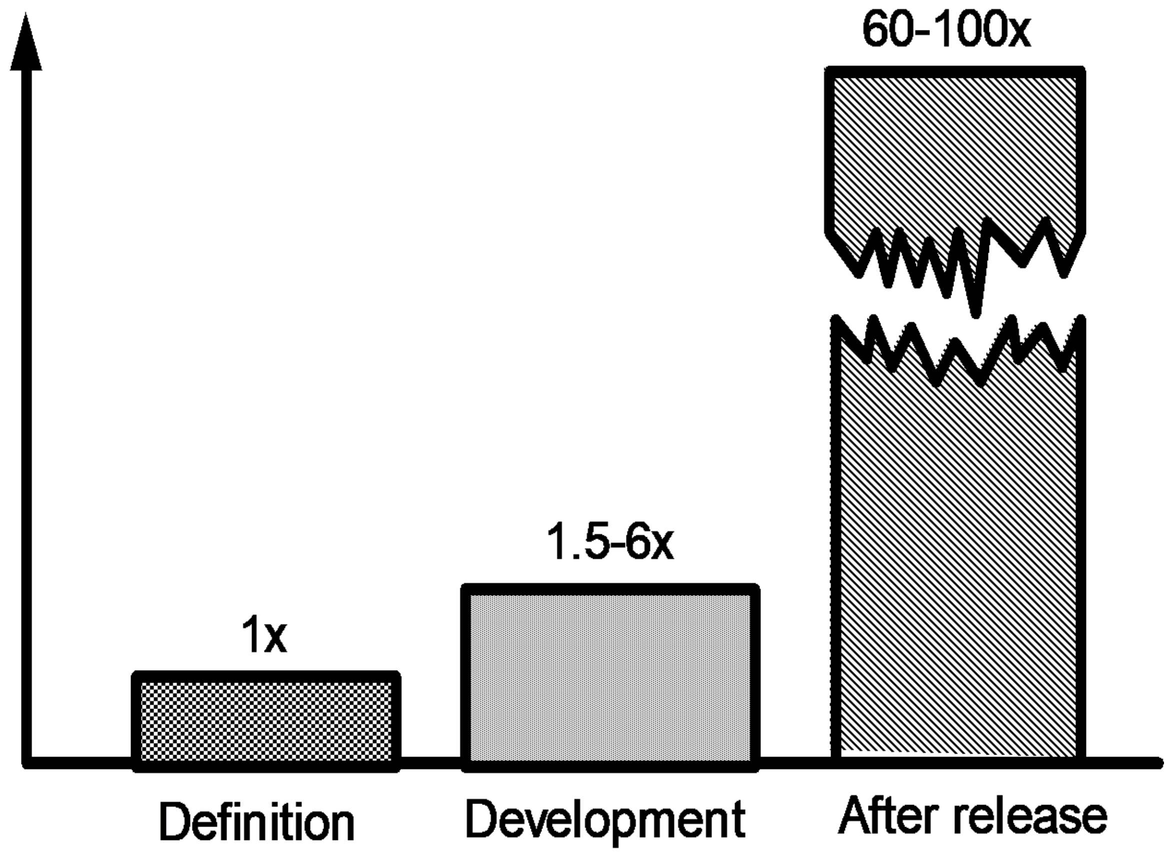
COST OF DETECTION & CORRECTION OF A FAULT



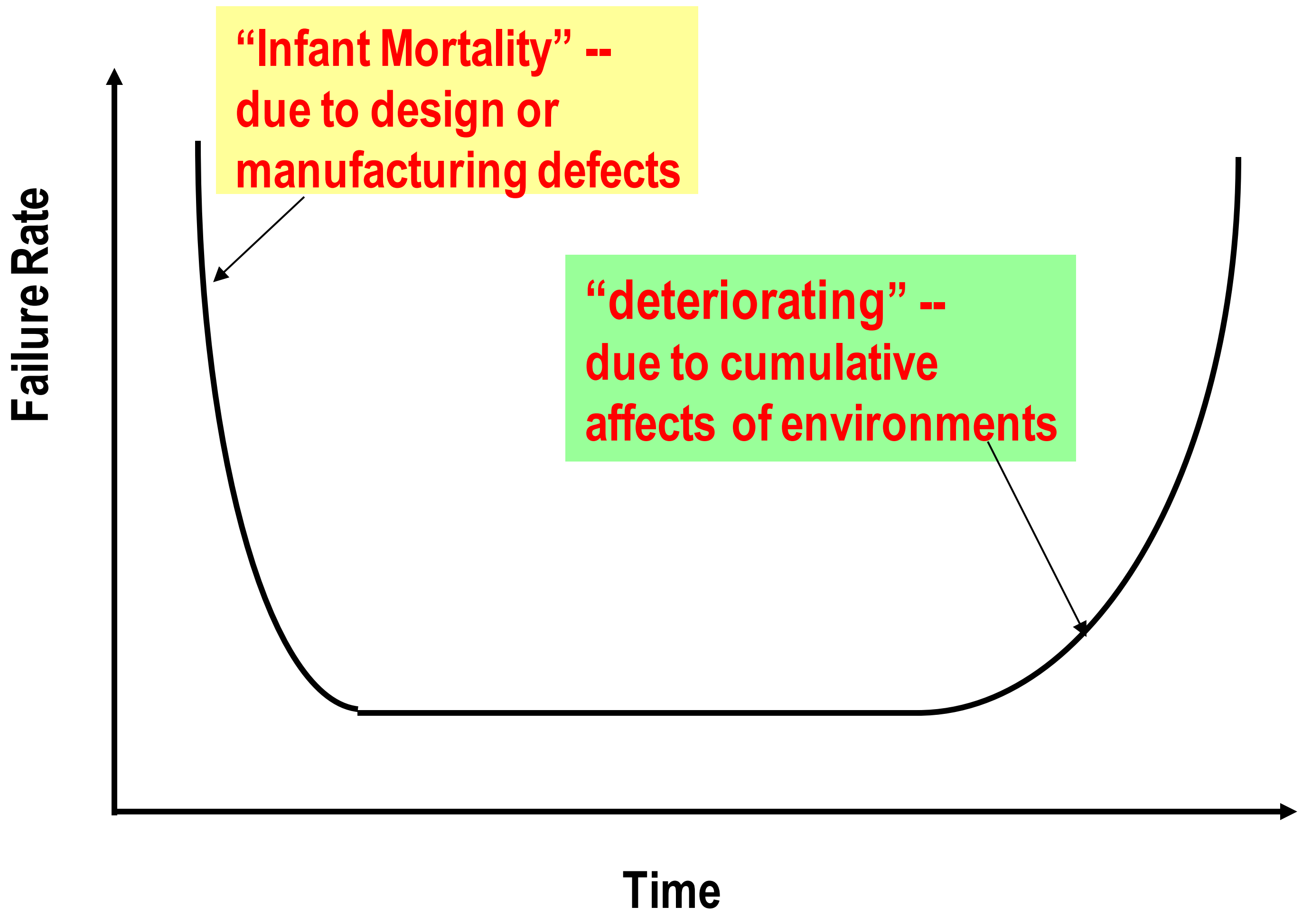
COST OF DETECTION & CORRECTION OF A FAULT



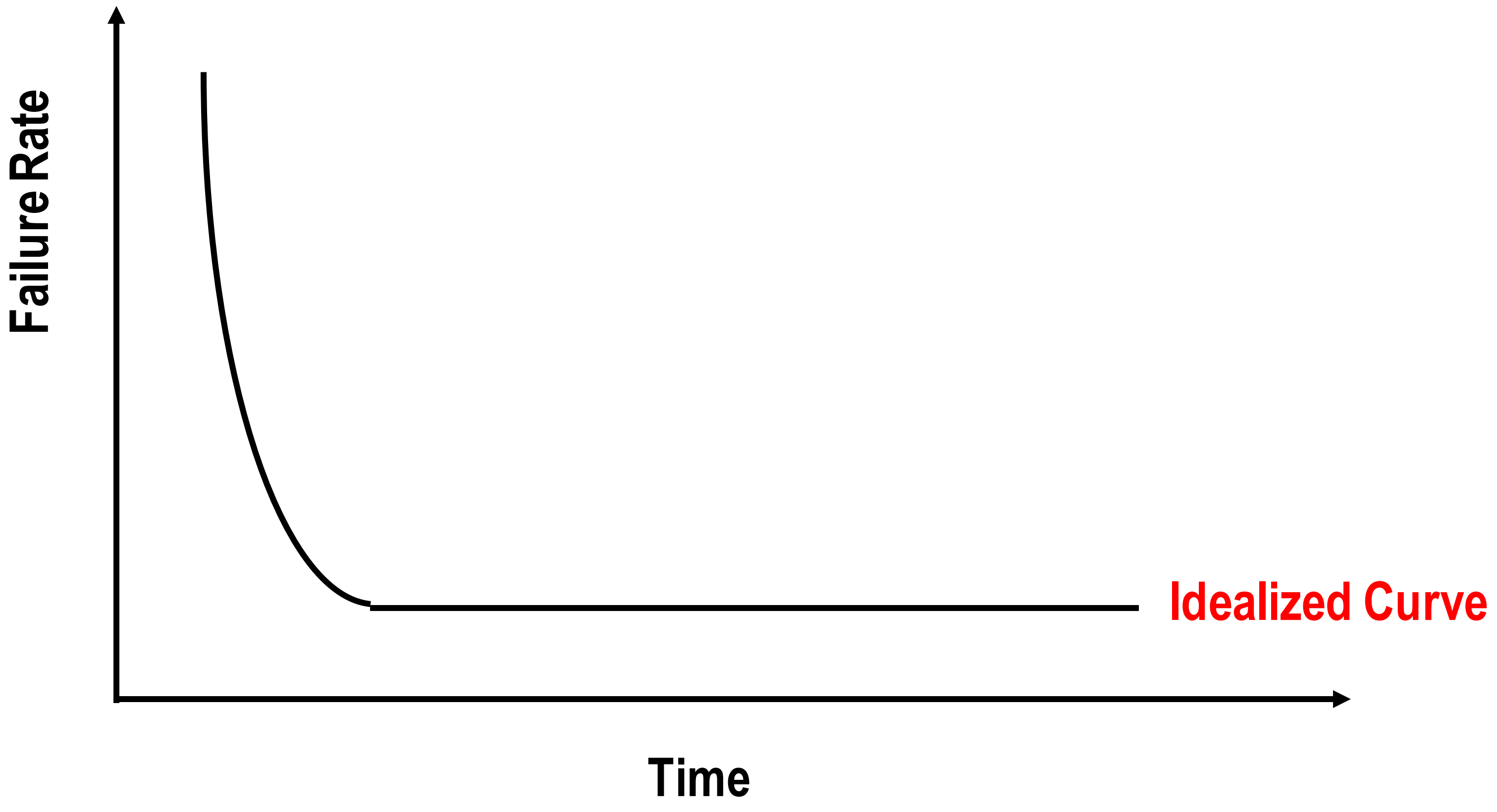
COST OF CHANGE



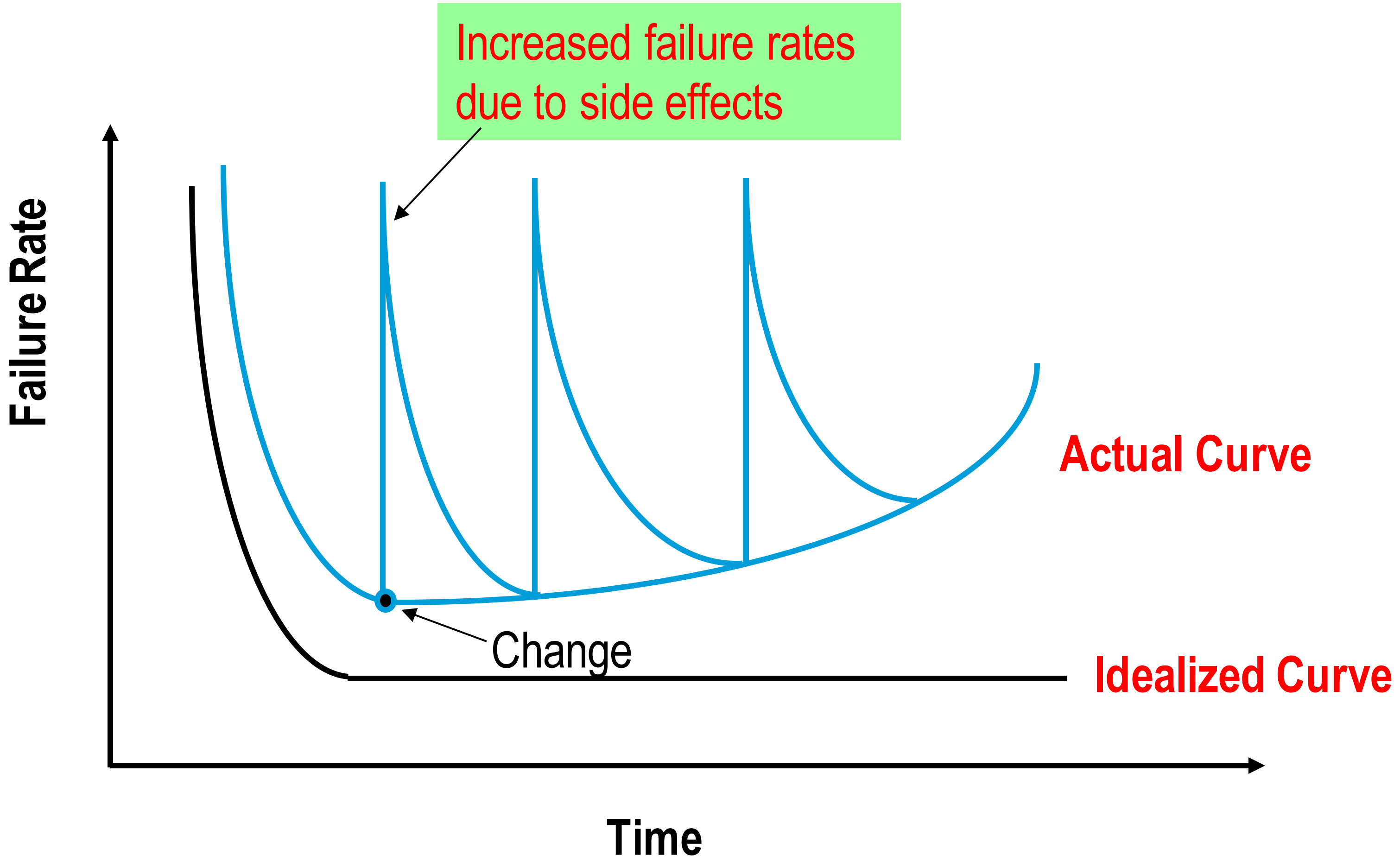
PRODUCT BATHTUB CURVE MODEL



SOFTWARE IDEALIZED CURVE



SOFTWARE ACTUAL FAILURE CURVE



WHAT IS SOFTWARE ENGINEERING?

- Technologies that make it easier, faster, and less expensive to build high-quality computer programs
- A discipline aiming to the production of fault-free software, delivered on time and within budget, that satisfies the users' needs

- An engineering: set of activities in software production
- The philosophy and paradigm of established engineering disciplines to solve what are termed software crisis

SOFTWARE APPLICATION

- System software (control computer H/W such as OS)
- Business software (commercial application for business users, SAP, ERP)
- Engineering and scientific software (e.g. statistical analysis-SPSS, matlab)
- Embedded software (e.g. auto pilot, biometric device)
- Personal computer software (e.g. Microsoft Office)
- Web-based software (use over internet with browser, e.g. Gmail)
- Artificial intelligence software (interact with computer, HCI, game)

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: My people have state-of-the-art software development tools, after all, we buy them the newest computers.
- Myth3: If we get behind schedule, we can add more programmers and catch up.
- Myth4: 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.

REFERENCES

- R.S. Pressman & Associates, Inc. (2010). *Software Engineering: A Practitioner's Approach.*
- Kelly, J. C., Sherif, J. S., & Hops, J. (1992). An analysis of defect densities found during software inspections. *Journal of Systems and Software*, *17*(2), 111-117.
- Bhandari, I., Halliday, M. J., Chaar, J., Chillarege, R., Jones, K., Atkinson, J. S., & Yonezawa, M. (1994). In-process improvement through defect data interpretation. *IBM Systems Journal*, *33*(1), 182-214.