An Introduction to Software Engineering

Software Engineering, Ian Summerville,

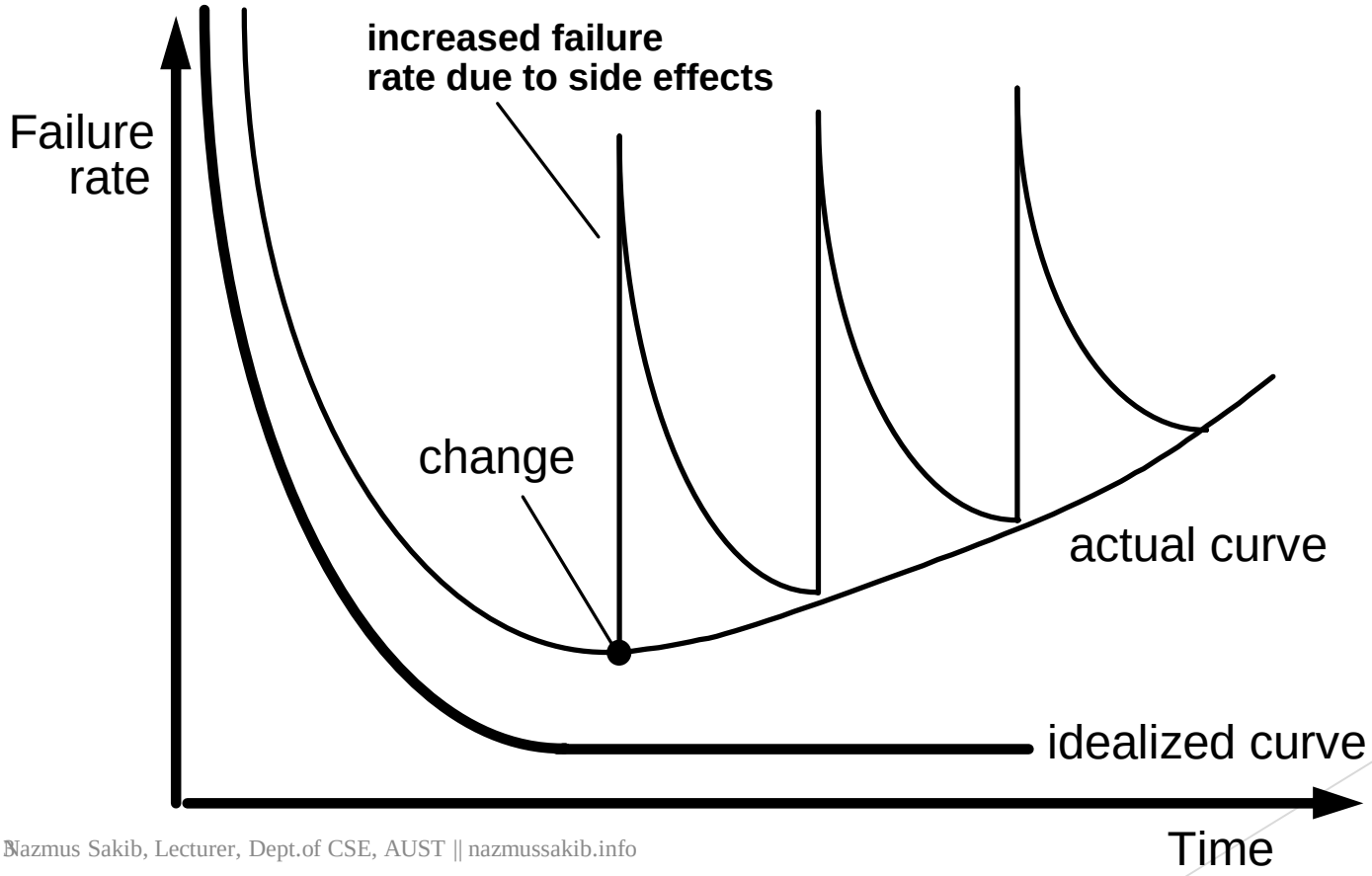
Lecture 02

Nazmus Sakib
Lecturer, Dept. of CSE, AUST

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

Wear vs. Deterioration



FAQs about software engineering

- What is software?
- What is software engineering?
- What is the difference between software engineering and computer science?
- What is the difference between software engineering and system engineering?
- What is a software process?
- What is a software process model?

FAQs about software engineering

- What are the costs of software engineering?
- What are software engineering methods?
- What is CASE (Computer-Aided Software Engineering)
- What are the attributes of good software?
- What are the key challenges facing software engineering?

What is software?

- Computer programs and associated documentation such as requirements, design models and user manuals.
- Software products may be developed for a particular customer or may be developed for a general market.
- Software products may be
  - Generic - developed to be sold to a range of different customers e.g. PC software such as Excel or Word.
  - Bespoke (custom) - developed for a single customer according to their specification.
- New software can be created by developing new programs, configuring generic software systems or reusing existing software.

What is software engineering?

- Software engineering is an engineering discipline that is concerned with all aspects of software production.
- Software engineers should adopt a systematic and organised approach to their work and use appropriate tools and techniques depending on the problem to be solved, the development constraints and the resources available.

What is the difference between software engineering and computer science?

- Computer science is concerned with theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.
- Computer science theories are still insufficient to act as a complete underpinning for software engineering (unlike e.g. physics and electrical engineering).

What is the difference between software engineering and system engineering?

- System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is part of this process concerned with developing the software infrastructure, control, applications and databases in the system.
- System engineers are involved in system specification, architectural design, integration and deployment.

What is a software process?

- A set of activities whose goal is the development or evolution of software.
- Generic activities in all software processes are:
  - Specification - what the system should do and its development constraints
  - Development - production of the software system
  - Validation - checking that the software is what the customer wants
  - Evolution - changing the software in response to changing demands.

What is a software process model?

- A simplified representation of a software process, presented from a specific perspective.
- Examples of process perspectives are
  - Workflow perspective - sequence of activities;
  - Data-flow perspective - information flow;
  - Role/action perspective - who does what.
- Generic process models
  - Waterfall;
  - Iterative development;
  - Component-based software engineering.

What are the costs of software engineering?

- Roughly 60% of costs are development costs, 40% are testing costs. For custom software, evolution costs often exceed development costs.
- Costs vary depending on the type of system being developed and the requirements of system attributes such as performance and system reliability.
- Distribution of costs depends on the development model that is used.

Software Applications

- system software
- application software
- engineering/scientific software
- embedded software
- product-line software
- WebApps (Web applications)
- AI software

Software—New Categories

- Open world computing—pervasive, distributed computing
- 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 31)
  - Data mining
  - Grid computing
  - Cognitive machines
  - Software for nanotechnologies

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 are software engineering methods?

- Structured approaches to software development which include system models, notations, rules, design advice and process guidance.
- Model descriptions
  - Descriptions of graphical models which should be produced; e.g. Object/data flow/state machine models etc.
- Rules
  - Constraints applied to system models; e.g. Unique name for every entity
- Recommendations
  - Advice on good design practice; e.g. No more than seven sub-objects associated with a single object
- Process guidance
  - What activities to follow; e.g. Object attributes documented before defining the operations associated with an object.

What is CASE (Computer-Aided Software Engineering)

- Software systems that are intended to provide automated support for software process activities.
- CASE systems are often used for method support.
- Upper-CASE
  - Tools to support the early process activities of requirements and design;
- Lower-CASE
  - Tools to support later activities such as programming, debugging and testing.

What are the attributes of good software?

- The software should deliver the required functionality and performance to the user and should be maintainable, dependable and acceptable.
- Maintainability
  - Software must evolve to meet changing needs;
- Dependability
  - Software must be trustworthy;
- Efficiency
  - Software should not make wasteful use of system resources;
- Acceptability
  - Software must accepted by the users for which it was designed. This means it must be understandable, usable and compatible with other systems.

What are the key challenges facing software engineering?

- Heterogeneity, delivery and trust.
- Heterogeneity
  - Developing techniques for building software that can cope with heterogeneous platforms and execution environments;
- Delivery
  - Developing techniques that lead to faster delivery of software;
- Trust
  - Developing techniques that demonstrate that software can be trusted by its users.

Issues of professional responsibility

- Confidentiality
  - Engineers should normally respect the confidentiality of their employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.
- Competence
  - Engineers should not misrepresent their level of competence. They should not knowingly accept work which is outwith their competence.

Issues of professional responsibility

- Intellectual property rights
  - Engineers should be aware of local laws governing the use of intellectual property such as patents, copyright, etc. They should be careful to ensure that the intellectual property of employers and clients is protected.
- Computer misuse
  - Software engineers should not use their technical skills to misuse other people's computers. Computer misuse ranges from relatively trivial (game playing on an employer's machine, say) to extremely serious (dissemination of viruses).

Hooker's General Principles

- 1: *The Reason It All Exists*
- 2: *KISS (Keep It Simple, Stupid!)*
- 3: *Maintain the Vision*
- 4: *What You Produce, Others Will Consume*
- 5: *Be Open to the Future*
- 6: *Plan Ahead for Reuse*
- 7*: Think!*