Lecture 1

Software Engineering

Hoang Huu Hanh, PTIT
hoanghuuhanh@ptit·edu·vn

Why Software Engineering?

- **Software development is hard !**
- **Important to distinguish "simple" systems** (*one developer, one user, experimental use only*) **from "complex" systems** (*multiple developers, multiple users, products*)
- **Experience with simple systems is misleading**
  - *One person techniques do not scale up*
- **Analogy with bridge building:**
  - Over a stream = easy, one person job
  - Over Mekong River … ? (*the techniques do not scale*)

Why Software Engineering ?

- The problem is ***complexity***
- Many sources, but ***size*** is key:
  - UNIX contains 4 million lines of code
  - Windows 2000 contains 10^8 lines of code

Software engineering is about managing this complexity.

Outline Syllabus

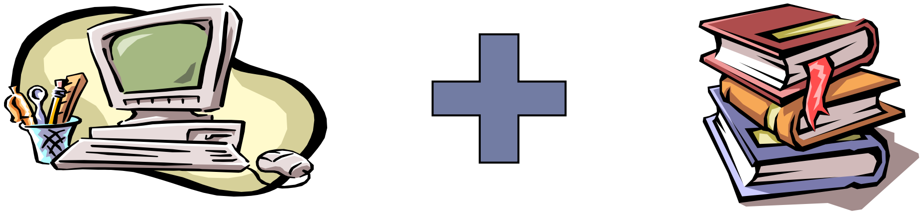
- Introduction to Software Engineering
- Software models
- Software requirements
- ~~Formal Specification~~
  - ~~*ASML (Abstract State Machines Language)*~~
- Software Design and Implementation
  - *UML (Unified Modeling Language)*
- Software verification, validation and testing
- Management of Software Projects & Cost Estimation

FAQs about software engineering

- What is
  - software?
  - software process?
  - software engineering?
  - software process model?

What is software?

- **Computer programs** and **associated documentation**



- **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
  - **Bespoke** (custom) - developed for a single customer according to their specification

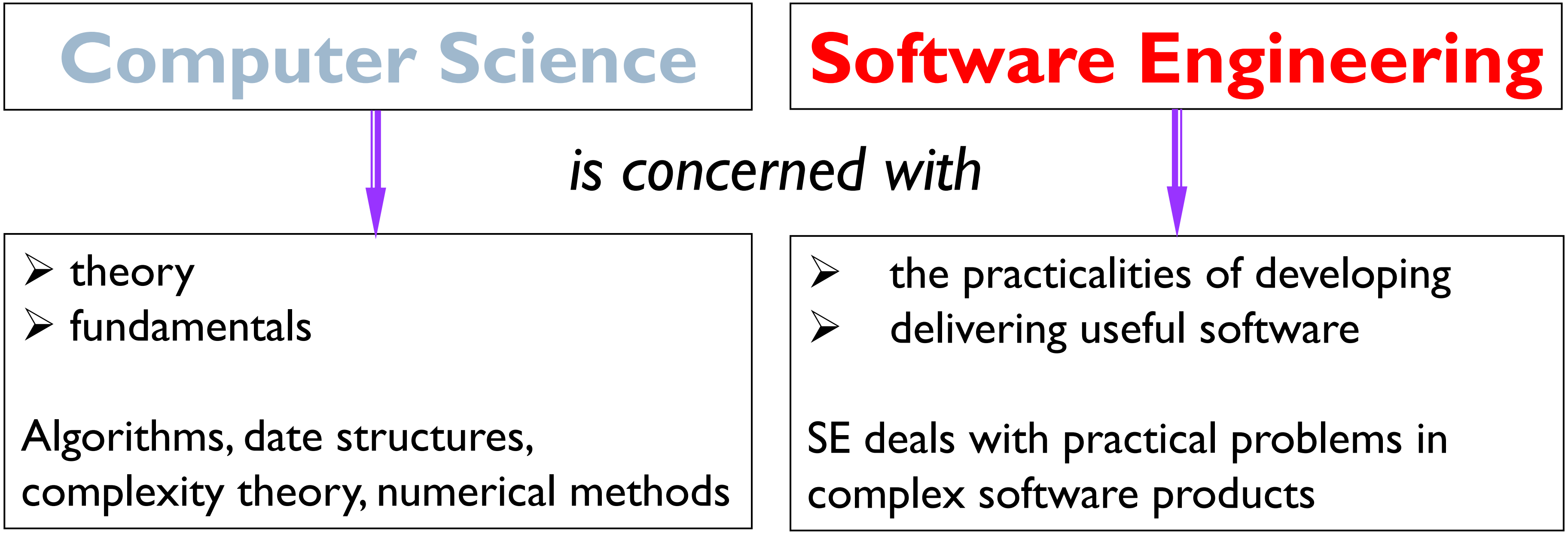
What is software engineering?

Software engineering is an engineering discipline which is concerned with all aspects of software production

Software engineers should

- adopt a systematic and organised approach to their work
- 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

Software Engineering

is concerned with

- theory
- fundamentals

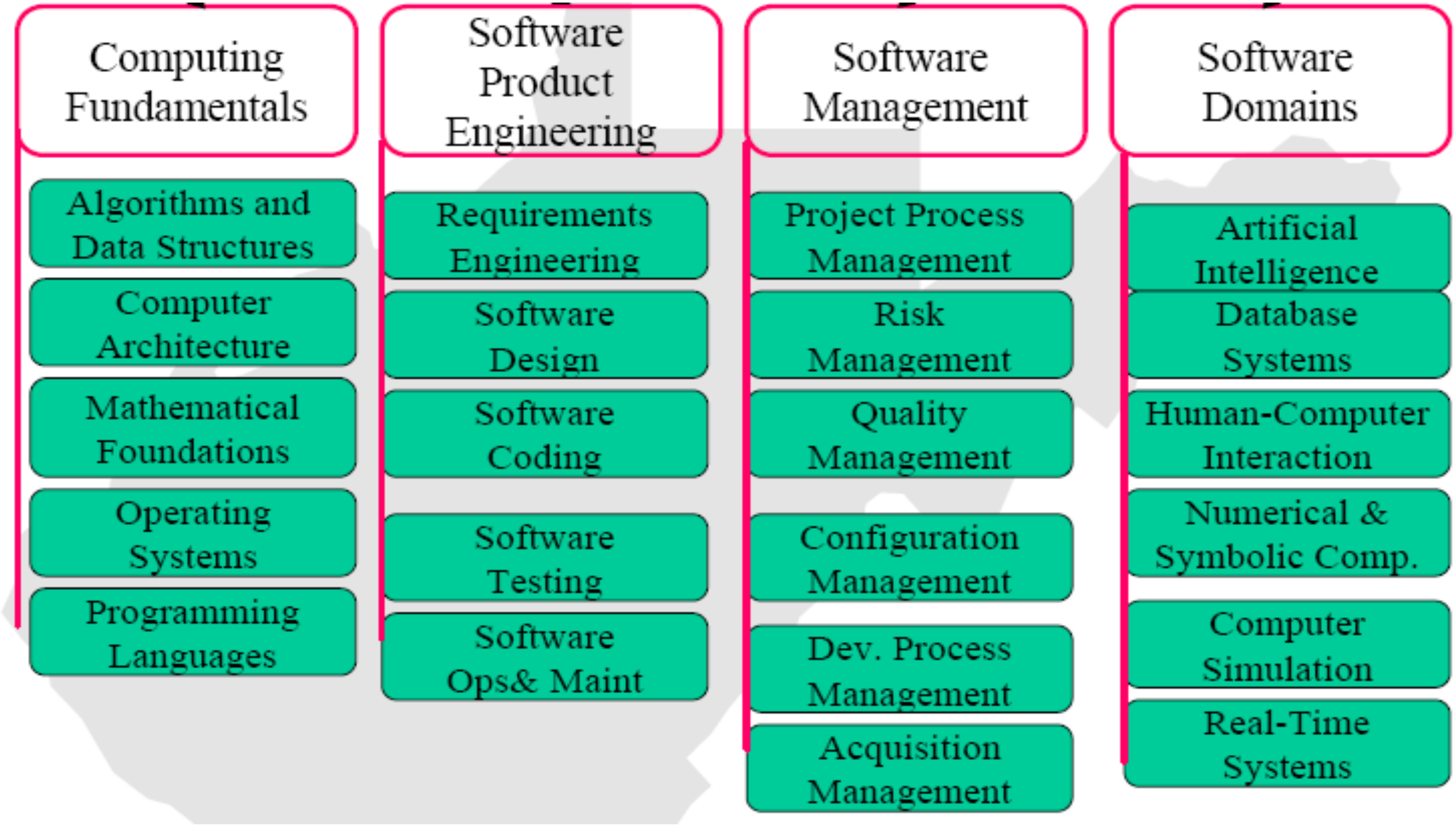
Algorithms, date structures, complexity theory, numerical methods

- the practicalities of developing
- delivering useful software

SE deals with practical problems in complex software products

Computer science theories are currently insufficient to act as a complete underpinning for software engineering, BUT it is a foundation for practical aspects of software engineering

Software Engineering Body of Knowledge



Computing Fundamentals

- Algorithms and Data Structures
- Computer Architecture
- Mathematical Foundations
- Operating Systems
- Programming Languages

Software Product Engineering

- Requirements Engineering
- Software Design
- Software Coding
- Software Testing
- Software Ops& Maint

Software Management

- Project Process Management
- Risk Management
- Quality Management
- Configuration Management
- Dev. Process Management
- Acquisition Management

Software Domains

- Artificial Intelligence
- Database Systems
- Human-Computer Interaction
- Numerical & Symbolic Comp.
- Computer Simulation
- Real-Time Systems

Source: http://www.sei.cmu.edu/pub/documents/99.reports/pdf/99tr004.pdf

SE history

- SE introduced first in 1968 – conference about "software crisis" when the introduction of third generation computer hardware led more complex software systems then before
- Early approaches based on informal methodologies leading to
  - Delays in software delivery
  - Higher costs than initially estimated
  - Unreliable, difficult to maintain software
- Need for new methods and techniques to manage the production of complex software.

Software myths

- **Management myths**
  - *Standards and procedures for building software*
  - *Add more programmers if behind the schedule*
- **Customer myths**
  - *A general description of objectives enough to start coding*
  - *Requirements may change as the software is flexible*
- **Practitioner myths**
  - *Task accomplished when the program works*
  - *Quality assessment when the program is running*
  - *Working program the only project deliverable*

Software failures

- **Therac-25 (1985-1987)**: six people overexposed during treatments for cancer
- **Taurus (1993)**: the planned automatic transaction settlement system for London Stock Exchange cancelled after five years of development
- **Ariane 5 (1996)**: roket exploded soon after its launch due error conversion (16 floating point into 16-bit integer)
- **The Mars Climate Orbiter** assumed to be lost by NASA officials (1999): different measurement systems (Imperial and metric)

However …

Important progress:

- Ability to produce more complex software has increased
- New technologies have led to new SE approaches
- A better understanding of the activities involved in software development
- Effective methods to specify, design and implement software have been developed
- New notations and tools have been produced

What is a software process?

- SP is a **set of activities** whose goal is the development or evolution of software
- Fundamental activities in all software processes are:
  - **Specification** - what the system should do and its development constraints
  - **Development** - production of the software system (design and implementation)
  - **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?

SPM is a simplified representation of a software process, presented from a specific perspective

- **Examples of process perspectives:**

  Workflow perspective represents inputs, outputs and dependencies

  Data-flow perspective represents data transformation activities

  Role/action perspective represents the roles/activities of the people involved in software process

- **Generic process models**
  - **Waterfall**
  - **Evolutionary development**
  - **Formal transformation**
  - **Integration from reusable components**

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**

What is **CASE ?**
(Computer-Aided Software Engineering)

Software systems which are intended to provide automated support for software process activities, such as requirements analysis, system modelling, debugging and testing

- **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 usable

- **Maintainability**
  - Software must evolve to meet changing needs
- **Dependability**
  - Software must be trustworthy
- **Efficiency**
  - Software should not make wasteful use of system resources
- **Usability**
  - Software must be usable by the users for which it was designed

What are the key challenges facing software engineering?

Software engineering in the 21st century faces three key challenges:

- **Legacy systems (các hệ thống để lại)**
  - Old, valuable systems must be maintained and updated
- **Heterogeneity (tính hỗn tạp)**
  - Systems are distributed and include a mix of hardware and software
- **Delivery (xuất xưởng)**
  - There is increasing pressure for faster delivery of software

Thank you!

Next lecture… ***Software Process***