Fundamental of Software Engineering
CSE 3205

Chapter- One
Introduction

School of Electrical Engineering and Computing
Department of computer Science and Engineering
ASTU

11 October 2017

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

Programs versus Software Products

• Usually small in size	• Large
• Author himself is sole user	• Large number of users
• Single developer	• Team of developers
• Lacks proper user interface	• Well-designed interface
• Lacks proper documentation	• Well documented & user-manual prepared
• Ad hoc development.	• Systematic development

I. What is Software Engineering?

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

  - ✓ Software engineering is a **modeling activity.**
  - ✓ Software engineering is a **problem-solving activity.**
  - ✓ Software engineering is a **knowledge acquisition activity.**
  - ✓ Software engineering is a **rationale-*driven activity*.**

Cont..

1. **Modeling Activity:**

- A model is an abstract representation of a system that enables us to answer questions about the system.
- Software engineers deal with complexity through modeling, by focusing at any one time on only the relevant details and ignoring everything else.
- Models are useful when dealing with systems that are too large, too small, too complicated, or too expensive to experience firsthand.

Cont…

2. Problem solving

- Engineering is a problem-solving activity. It is not algorithmic. In its simplest form, the engineering method includes five steps:
    1. formulate the problem
    2. analyze the problem
    3. search for solutions
    4. decide on the appropriate solution
    5. specify the solution

Cont…

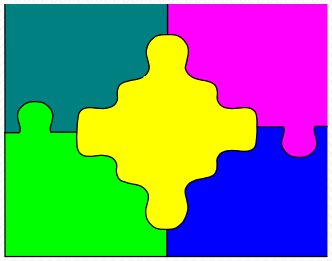
3. Knowledge acquisition

- In modeling the application and solution domain, software engineers collect data, organize it into information, and formalize it into knowledge.
- Knowledge acquisition is **nonlinear**, as a single piece of data can invalidate complete models.
- A common mistake that software engineers and managers make is to assume that the acquisition of knowledge needed to develop a system is linear.

Why Study Software Engineering? (1)

- To acquire skills to develop large programs.
  - Exponential growth in complexity and difficulty level with size.
  - The ad hoc approach breaks down when size of software increases .

Why Study Software Engineering? (2)



- Ability to solve complex programming problems:
  - How to break large projects into smaller and manageable parts?
- Learn techniques of:
  - specification, design, interface development, testing, project management, etc.

Why Study Software Engineering? (3)

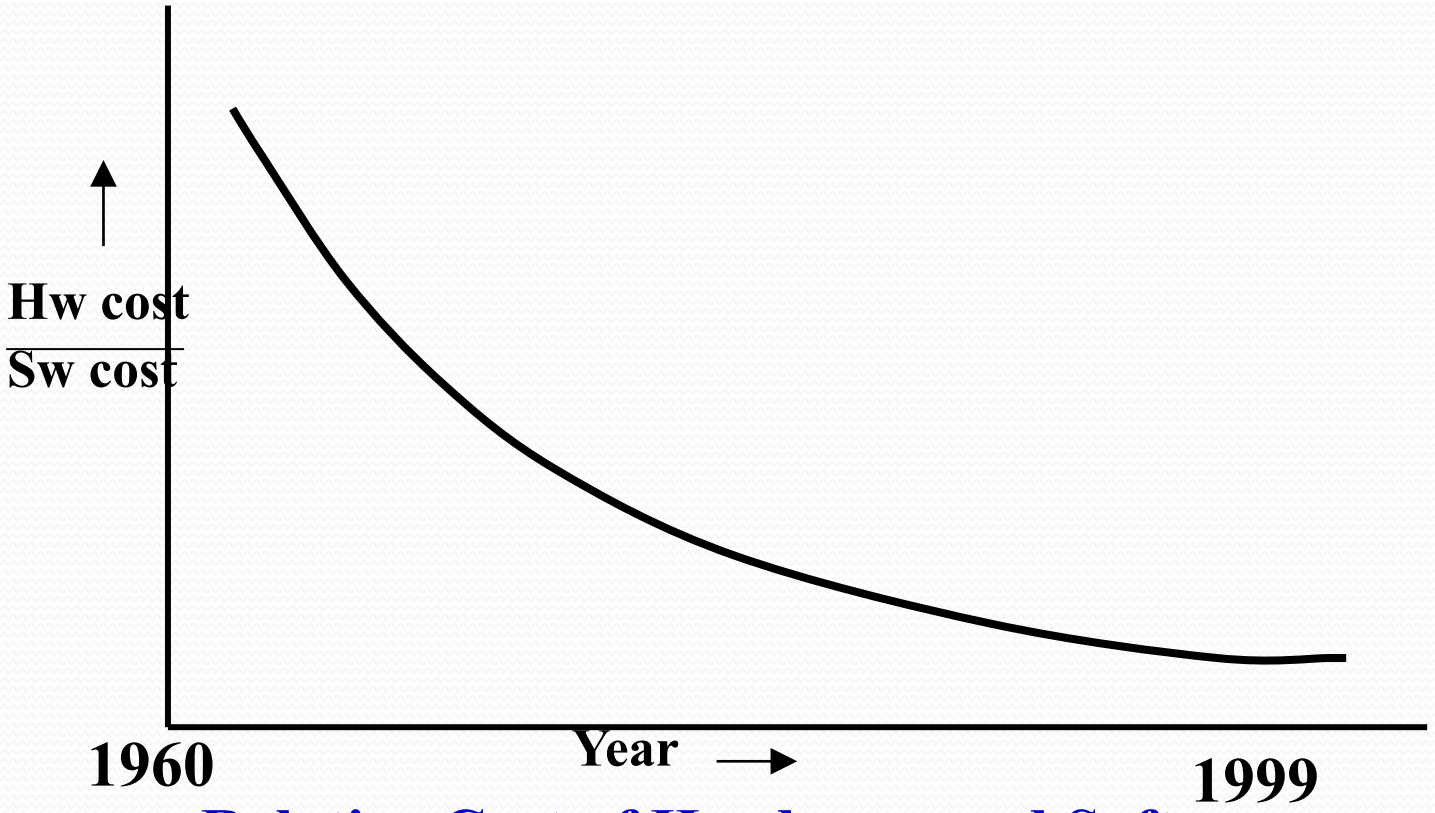
- To acquire skills to be a better programmer:
  - Higher Productivity
  - Better Quality Programs

Why Study Software Engineering? (4)

Software Crisis

- Software products:
  - fail to meet user requirements.
  - frequently crash.
  - expensive.
  - difficult to alter, debug, and enhance.
  - often delivered late.
  - use resources non-optimally.

Software Crisis (cont.)



Relative Cost of Hardware and Software

Why Study Software Engineering? (5)

- **1950s and 1960s, Systems Development was *unstructured* & *unorganized***
- **Leap-year bug**
  - ✓ A supermarket was fined $1000 for having meat around 1 day too long, on February 29, 1988.
- **Interface misuse**
  - ✓ On April 10, 1990, in London, an underground train left the station without its driver.
- **Security**
  - ✓ On November 2, 1988, a self-propagating program, subsequently called the Internet Worm, An estimated 10% of all Internet nodes were affected. The infection took several days to eradicate.

Cont…

- **Late and over budget**
  - ✓ In 1995, bugs in the automated luggage system of the new Denver International Airport caused suitcases to be chewed up. The airport opened 16 months late, $3.2 billion over-budget, with a mostly manual luggage system.
- **On-time delivery**
  - ✓ After 18 months of development, a $200 million system was delivered to a health insurance company in Wisconsin in 1984. However, the system did not work correctly: $60 million in overpayments were issued. The system took 3 years to fix.
- **Unnecessary complexity**
  - ✓ The C-17 cargo plane by McDonnell Douglas ran $500 million over budget because of problems with its avionics software. The C-17 included 19 onboard computers, 80 microprocessors, and 6 different programming languages.

Cont..

- SE introduced first in 1968 – conference about "software crisis" when the introduction of third generation computer hardware led more complex software systems than 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.

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

Cont…

4. **Rationale management**

- When acquiring knowledge and making decisions about the system or its application domain, software engineers also need to capture the context in which decisions were made and the rationale behind these decisions.
  - enables software engineers to understand the implication of a proposed change when revisiting a decision.

What is the difference between software engineering and computer science?

Computer Science is concerned with **Software Engineering**

- theory
- fundamentals

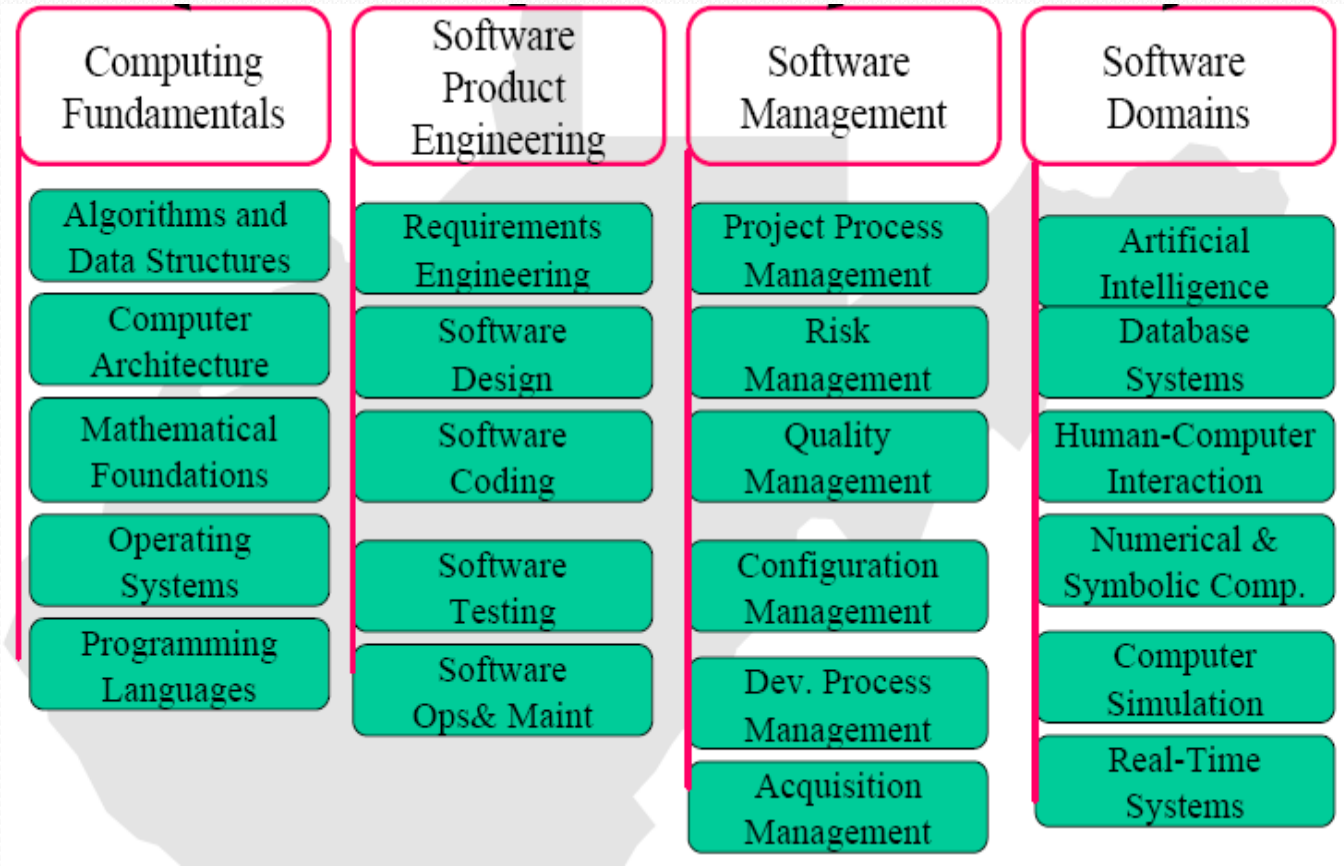
Algorithms, data structures, complexity theory, numerical methods

- the practicalities of developing and 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



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

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 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?

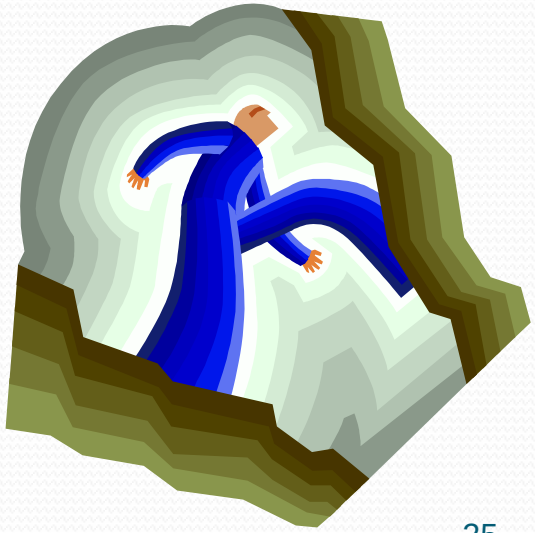
The software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable

- **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**
  - Old, valuable systems must be maintained and updated
- **Heterogeneity**
  - Systems are distributed and include a mix of hardware and software
- **Delivery**
  - There is increasing pressure for faster delivery of software



Thank You!

Q?