**Software Engineering**

# Quick Overview

Dr. Shihong Huang

Department of Computer Science & Engineering
Florida Atlantic University

## Lectures

- Course Overview
- ☞Introduction
- Management
- Requirements
- Design
- *Construction*
- Testing
- Maintenance & Evolution
- *Advanced Topics*
- Summary

## Objectives

- Learn why software engineering is an important discipline
- See why it is so hard to do properly
- Learn what software engineering – and what it is not
- Gain an appreciation for ethical and culpable software engineering

## Outline

- The challenges of software
- Software engineering
- The software process (brief introduction)
- Professionalism
- Summary

## Challenges of Software –1

**Software is everywhere**

- It makes up over 95% of the cost of a PC
- It’ s in nearly all embedded devices, including “information appliances”
- Most companies are now in the software business, whether they realize it or not
  - Aircraft
  - Cars
  - Finance
  - …

## Challenges of Software –2

**Software is Problematic**

- It’ s often late
- It’ s very expensive to construct
- It usually doesn’t meet requirements
- It’ s hard to understand
- It quickly becomes unmaintainable
- It’ s unreliable
- It’ s still inefficient – “bloatware”
- It’ s become far too complex

## Challenges of Software –3

**How Are We Doing? (1995)**

- 16% of software projects were finished on time & within budget
- Over 50% of projects cost nearly 200% of their original estimate
- Projects usually had less than 50% of originally proposed features/functions

## Challenges of Software –4

**How Are We Doing? (2002)**

- 75% of all software projects late or canceled
- 78% of IT organizations involved in disputes leading to litigation
- For the organizations that entered into litigation:
  - In 67% of the disputes, the functionality of the information system as delivered did not meet up to the claims of the developers
  - In 56% of the disputes, the promised delivery date slipped several times
  - In 45% of the disputes, the defects were so severe that the information system was unusable
- 50% of software is so defective / unusable
- Cost to U.S. economy: $5.85B

[Standish Group, Cutter Edge, NIST]

## Challenges of Software –5

**Software Failure is Expensive**

- American Airlines: $50M – schedule bug
- American Express: $167K per minute
- Charles Schwab: $1M per minute
- E*Trade, eBay, amazon.com, …

- Y2K: estimated at $600B (final bill= ?)

## Challenges of Software –6

**Ariane 5 Launcher Failure**

- A European rocket designed to launch commercial payloads into Earth orbit
- Successor to the successful Ariane 4 launchers
- 1996 June 4th, total failure Ariane 5's maiden flight
- Failure occurred at 37 seconds, altitude less than 4 kilometers
- Cost of failure: $500M

## Software Engineering –1

- Term “software engineering” first appeared at a NATO conference in Germany in 1968
- The IEEE computer society defines software engineering as “*the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; That is, <u>the application of engineering to software</u>.*” [IEEE standards guide, 1990]

## Software Engineering –2

- Based on computer science theory & practice
- 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).
- Theory:
  - Knowing which data structures to use
  - Knowing how to develop new algorithms
  - Understanding computability
- Practice:
  - Skilled at programming in various languages
  - Knowledgeable about computer architectures
  - Adept at engineering complex systems

## Software Engineering –3

- Currently, software engineering is neither a science nor an engineering discipline; it's more like a guild
- It is not (currently) a profession in the same sense as, say, electrical engineering
- It is not a profession like law or medicine

- The "in-the-large" companion to software construction
- It's not just coding…

## Software Engineering –4

| **Software Engineering** | **Software Construction** |
|---|---|
| Multi-person teams | Single programmer |
| Modifying an existing ill-defined system | Designing a clearly-specified system |
| Fixing bugs, other "post-deployment" activities | New "green field" development |
| Larger applications | Smaller applications |
| Professional | Hacker |

## Software Engineering –5

**Products**

- Different products have different goals:
  - Shrink-wrapped PC office automation software
  - Enterprise applications (CRM)
  - Space shuttle control software
- Software engineering is not just about producing products, but producing products in a cost-effective and predictable manner
- The goal is to produce high-quality software with a finite amount of resources on schedule

## Software Engineering –6

**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.

## Software Engineering –7

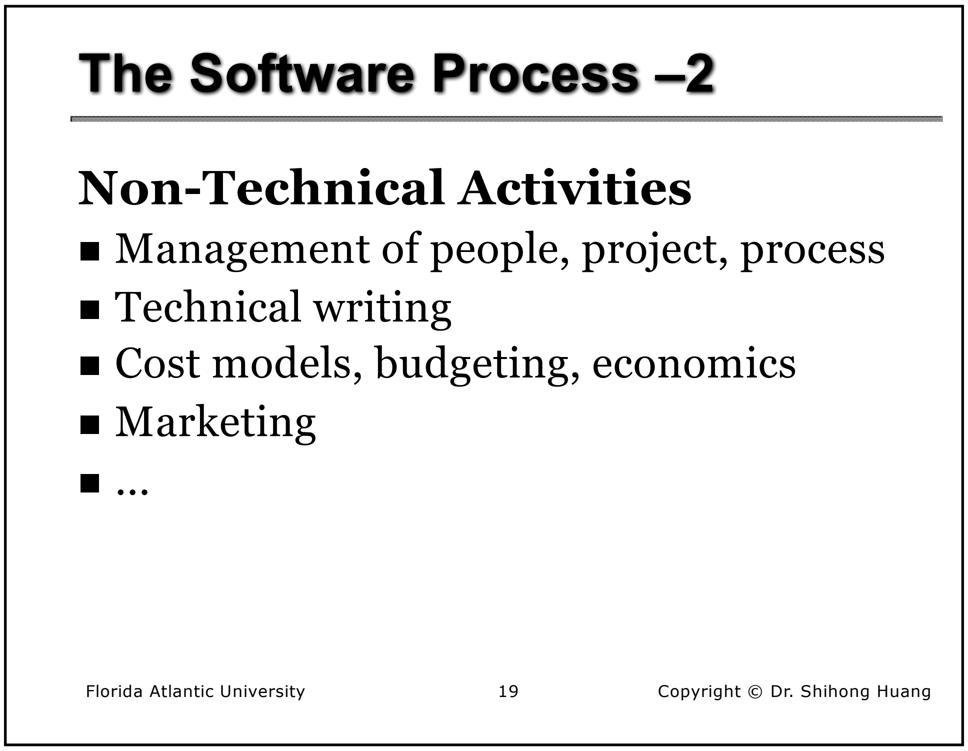
### Quality Attributes

- 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 be accepted by the users for which it was designed. This means it must be understandable, usable and compatible with other systems
- Usability:
- Security

- Assessing quality attributes is difficult

## The Software Process –1

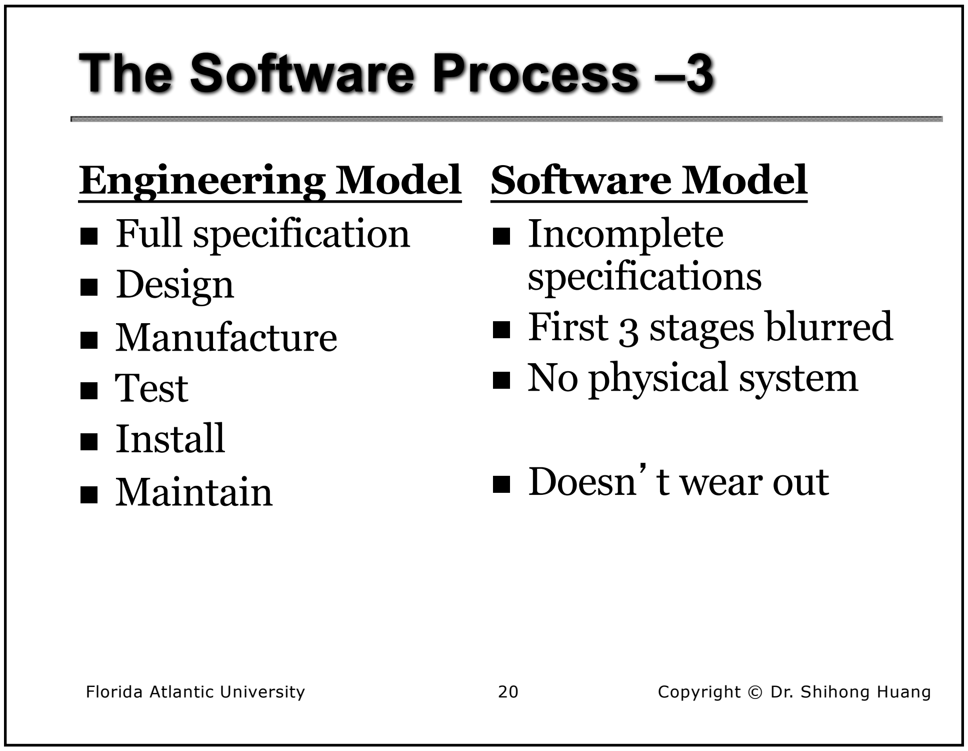
- Structured set of activities required to engineer a complex software system
- Fundamental process activities:
  - Requirements
  - Design
  - Construction
  - Testing
  - Maintenance and Evolution
- A process model is a software product



## The Software Process –2

**Non-Technical Activities**

- Management of people, project, process
- Technical writing
- Cost models, budgeting, economics
- Marketing
- …



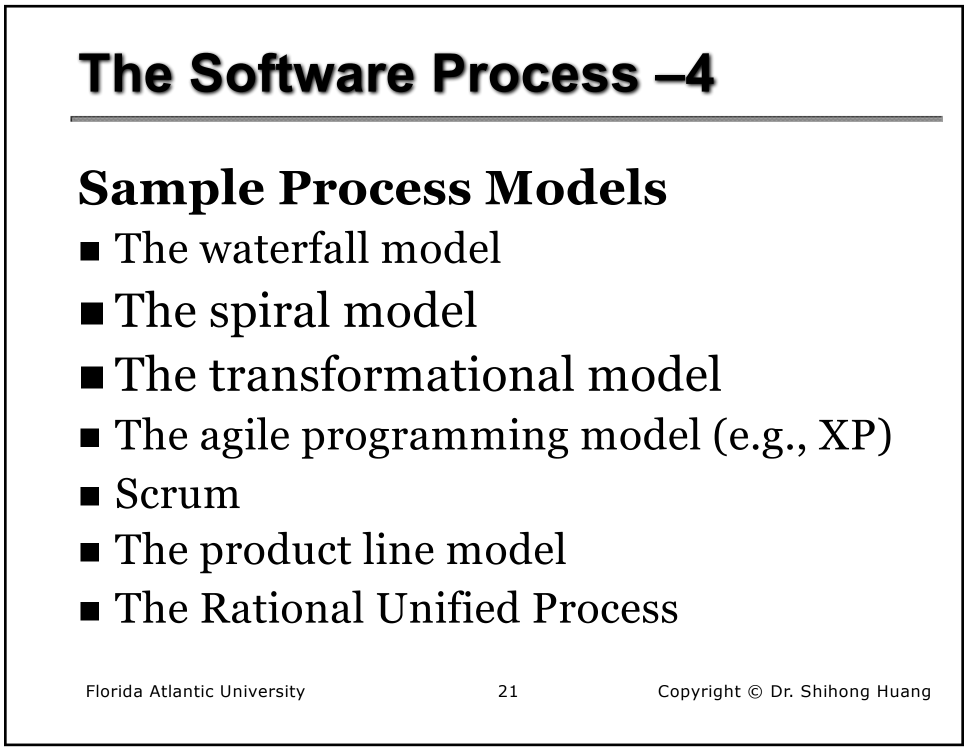
## The Software Process –3

**Engineering Model**

- Full specification
- Design
- Manufacture
- Test
- Install
- Maintain

**Software Model**

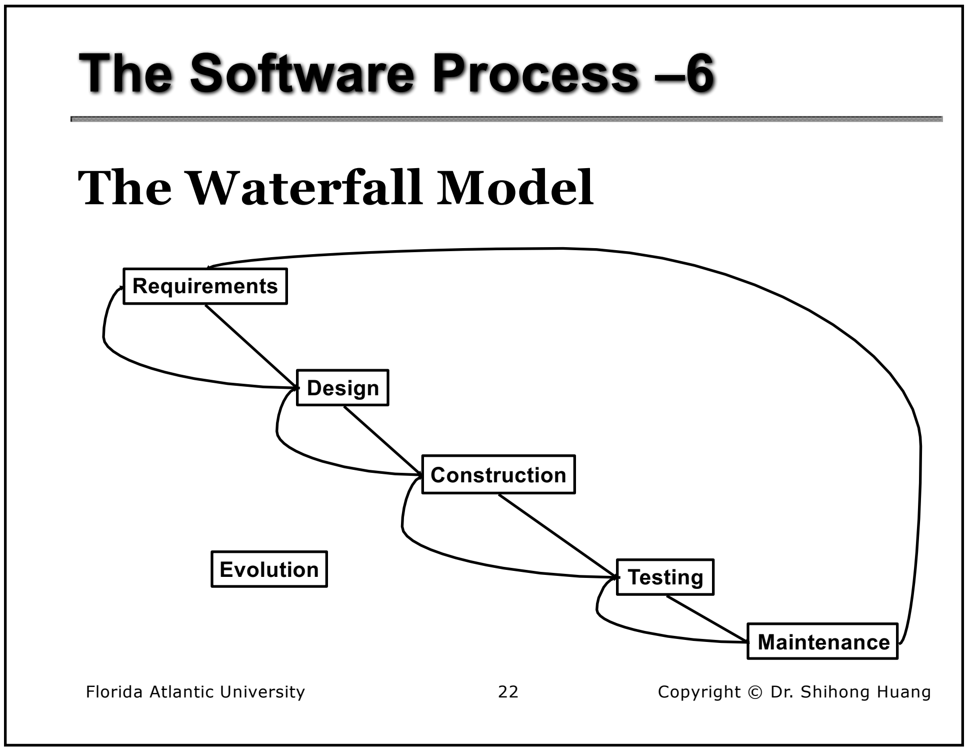
- Incomplete specifications
- First 3 stages blurred
- No physical system
- Doesn’t wear out



## The Software Process –4

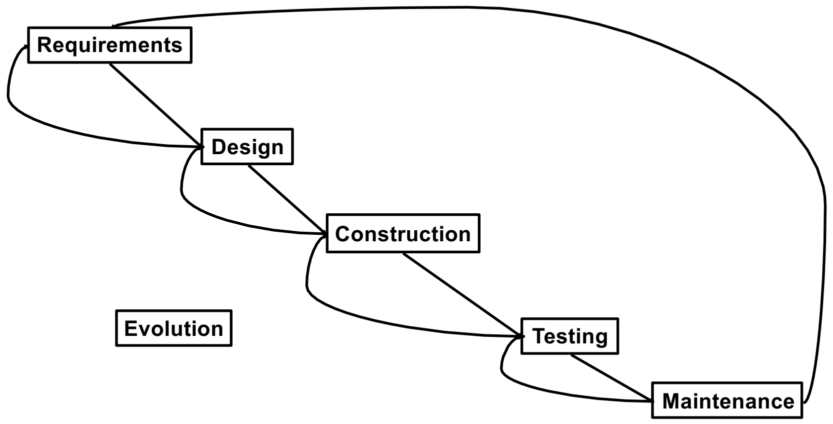
**Sample Process Models**

- The waterfall model
- The spiral model
- The transformational model
- The agile programming model (e.g., XP)
- Scrum
- The product line model
- The Rational Unified Process



## The Software Process –6

**The Waterfall Model**



## Professionalism –1

- Software is more than technical issues: ethical, social, professional, ..
- Personal choices:
  - Military systems development
  - Chicken little (cf. Y2K)
  - Community involvement
- Developers and maintainers need to be:
  - Hardworking
  - Intelligent
  - Sensible
  - Up-to-date, and above all..
  - Ethical

## Professionalism –2

- Software engineering involves wider responsibilities than simply the application of technical skills
- Software engineers must behave in an honest and ethically responsible way if they are to be respected as professionals
- Ethical behaviour is more than simply upholding the law

## Professionalism –3

- 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 beyond their competence
- 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).

## Professionalism –4

**Code of Ethics**

- IEEE-CS ACM Software Engineering Code of Ethics and Professional Practice (www.computer.org/tab/seprof/code.htm)
- Members of these organisations sign up to the code of practice when they join
- The Code contains eight principles related to the behaviour of and decisions made by professional software engineers, including practitioners, educators, managers, supervisors and policy makers, as well as trainees and students of the profession

## Professionalism –5

- SWEBOK: An attempt to codify the "Software Engineering Body of Knowledge"
- Software Engineering Education Knowledge (SEEK): A companion to SWEBOK, focusing on software engineering education

## Summary –1

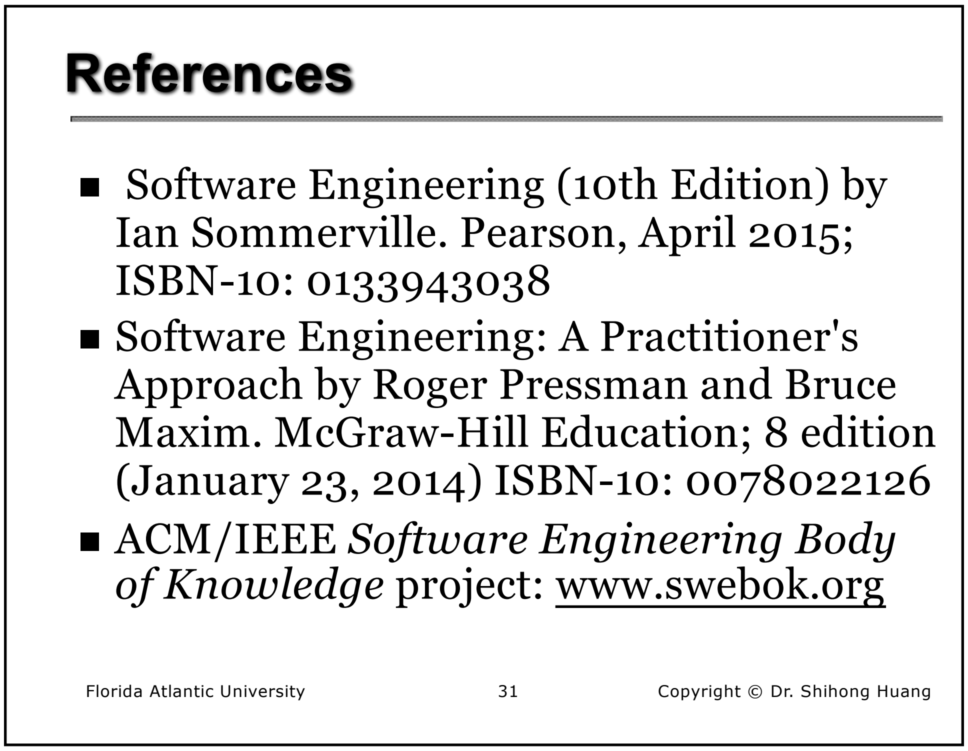
- Software is a critical and complex part of everyday life that impacts everyone
- Software engineering is a young, inter-disciplinary, and evolving field of study
- Technical and non-technical aspects are equally important to a successful project

## Summary –2

- The software process includes all activities of software engineering (not just coding)
- There are several process models, sharing the common and essential activities of requirements, design, construction, testing, and maintenance

## Summary –3

- Software engineering is not (yet) a generally recognized profession, but there are a code of ethics to follow
- To be a good software engineer, you need:
  - Inquisitive problem solving skills
  - To like to take things apart, to see what "makes them tick"
  - Agility and self-discipline



## References

- Software Engineering (10th Edition) by Ian Sommerville. Pearson, April 2015; ISBN-10: 0133943038
- Software Engineering: A Practitioner's Approach by Roger Pressman and Bruce Maxim. McGraw-Hill Education; 8 edition (January 23, 2014) ISBN-10: 0078022126
- ACM/IEEE *Software Engineering Body of Knowledge* project: www.swebok.org