CMPS310 Software Engineering
Fall 2021

Lecture 1

Introduction to Software Engineering

Topics covered

- What is software engineering
- Importance of software engineering
- Types of software
- Differences between software engineering and other similar disciplines
- Challenges in software engineering
- Professional and ethics related to software engineering
- Software quality
- Stakeholders of software engineering

What is Software?

- The product that software professionals build and then support over the long term
- Computer programs and associated documentations such as requirements, design models, architectural artefact, program and user manuals
- Software encompasses

1. Instructions (computer programs) that when executed provide desired features, function, and performance
2. Data structures that enable the programs to adequately store and manipulate information (*data type, abstract data type, etc.*)
3. Documentation that describes the operation and use of the programs (*user manual, specifications, design artifact,* etc.)

Software engineering – what is it?

- **Software engineering** is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use
- Engineering discipline
  - Using appropriate theories and methods to solve problems bearing in mind organizational and financial constraints
- All aspects of software production
  - Not just technical process of development
  - Also, project management and the development of tools, methods etc. to support software production
- **IEEE definition**: The application of a systematic, disciplined, quantifiable approach to the development, operation, maintenance of software; that is, the application of engineering to software

Software product category

- Software products may be developed for
  - a particular customer or a general market
- Software products may be
  - **Generic software**
    - Developed to be sold to a range of different customers
    - Stand-alone systems marketed and sold to any customer who wishes to buy them
    - Examples – PC software such as graphics programs, project management tools; CAD software; software for specific markets such as appointments systems for dentists
  - **Customized software**
    - Developed for a single customer according to their specification
    - Software that is commissioned by a specific customer to meet their own needs
    - Examples – embedded control systems, air traffic control software, traffic monitoring systems
  - **Software development from scratch**
    - New software can be created by developing new programs, configuring generic software systems or reusing existing software

Types of software

- ✧ Real time software
  - Must react immediately to data/environment
  - Safety often a concern
  - React immediately to its environment
  - Example, aircraft auto pilot system
- ✧ Data processing software
  - Used to run businesses
  - Accuracy and security of data are key
  - Example, QU student system
- ✧ *Some software has both aspects*

Main phases (tasks) of software development process

1. ***Requirements Elicitation (gathering information)***

   Define what the system must do and the constraints on its operation

2. ***Analysis*** (answers "***WHAT***?")

   Understand what's needed and produce a **conceptual domain model**

   = identify use cases + key concepts and their associations & attributes

3. ***Design*** (answers "***HOW***?")

   Specify the system components and how they will work together

4. ***Implementation*** ("***CODING***")

   Write the code = Translate the design into running software

5. ***Testing*** (type of ***VERIFICATION***)

   Verify that the resulting software meets the requirements

6. ***Maintenance*** (***REPAIR*** or ***ENHANCEMENT***)

   Repair defects and add enhancements to meet changing requirements

Example of software development process

- ***Requirements*** => Text produced

  e.g., " … The application shall display the balance in the user's bank account. …"
- ***Design*** => Diagrams and text

  e.g., " … The design will consist of the classes *CheckingAccount*, *SavingsAccount*, …"
- ***Implementation*** => Source code

  e.g., … class CheckingAccount { double balance; … } …
- ***Testing*** => Test cases and test results

  e.g., "… With test case: *deposit $44.92 / deposit $32.00 / withdraw $101.45 /* … the balance was $2938.22, which is correct. …"
- ***Maintenance*** => Modified design, code, and test

  e.g., Defect repair: "Application crashes when balance is $0 and attempt is made to withdraw funds. …"

  e.g., Enhancement: "Send SMS message to Customer when balance changes"

Analysis vs. Design?

Analysis

investigation

What?

- Includes
  1. Requirements analysis
  2. Domain analysis
- Key Questions
  - How the system will be used (i.e., use cases)?
  - Finding and describing the key concepts – or objects – in the problem domain as well as their attributes and associations (i.e., **conceptual domain model**)

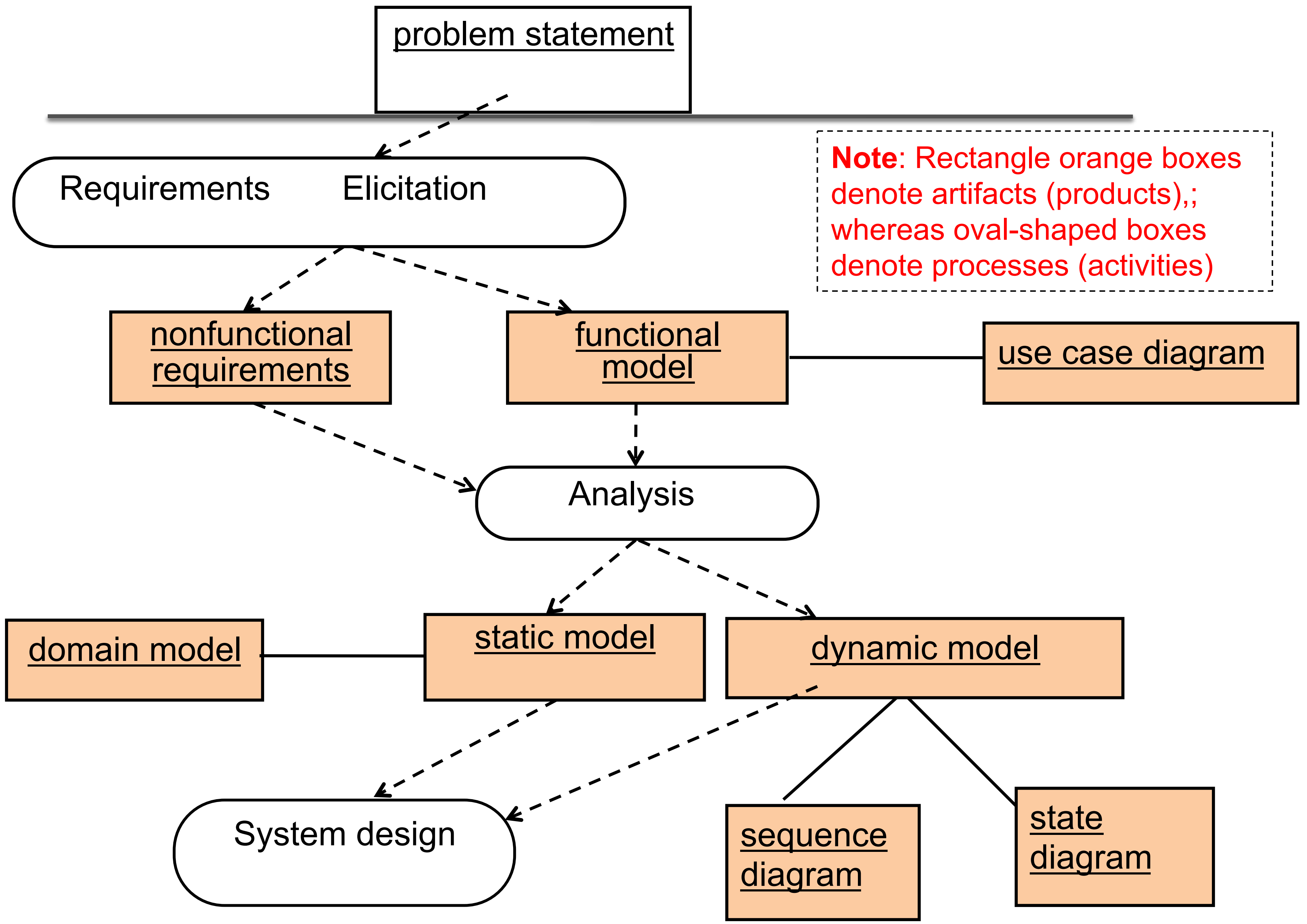
Design

solution

How?

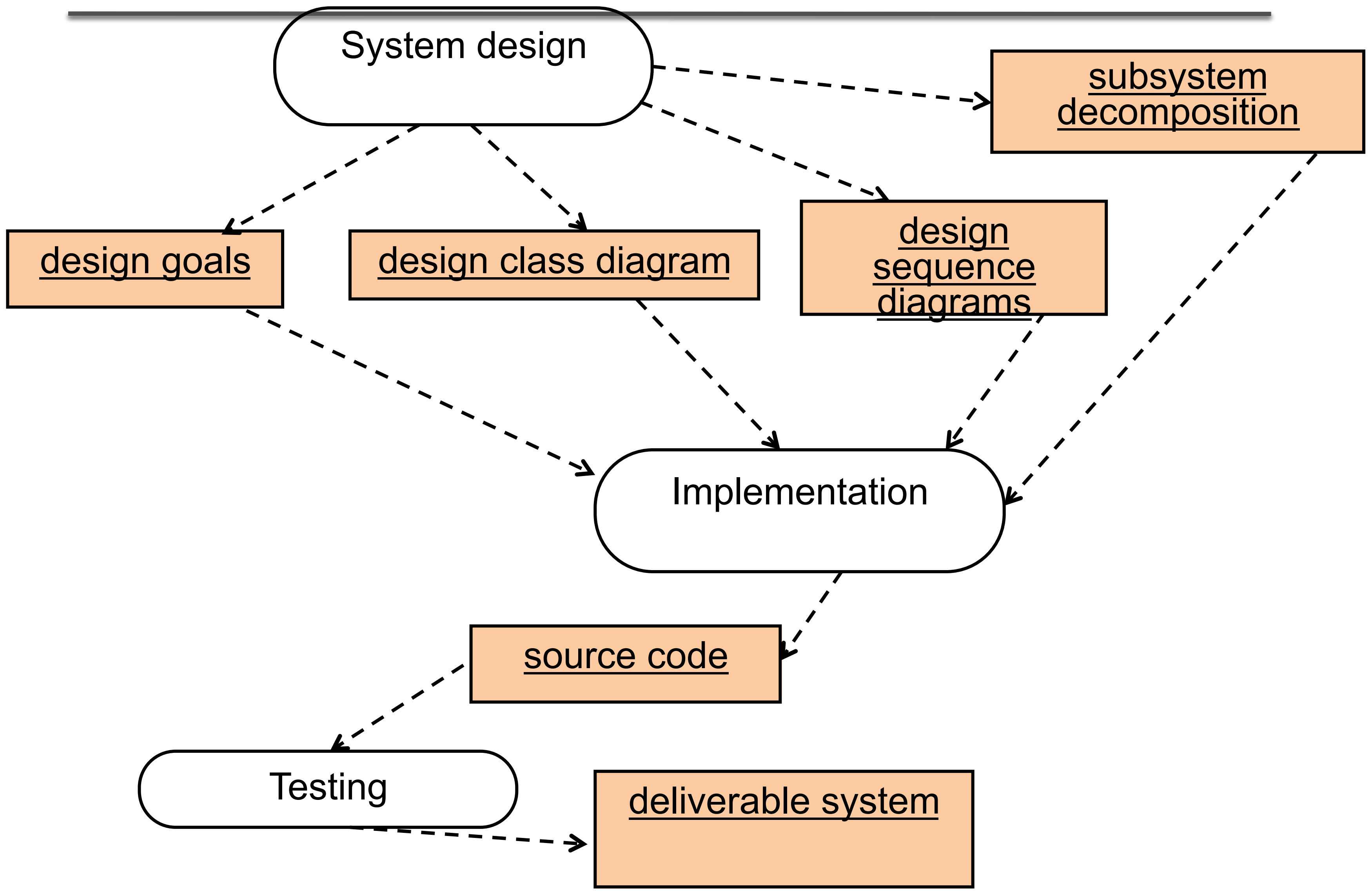
- Includes
  1. Object design
  2. Database design
  3. User Interface (UI) design
- Key Questions
  - How should responsibilities be assigned to classes?
  - How should objects interact to fulfill the requirements?

OO modeling in software engineering process I

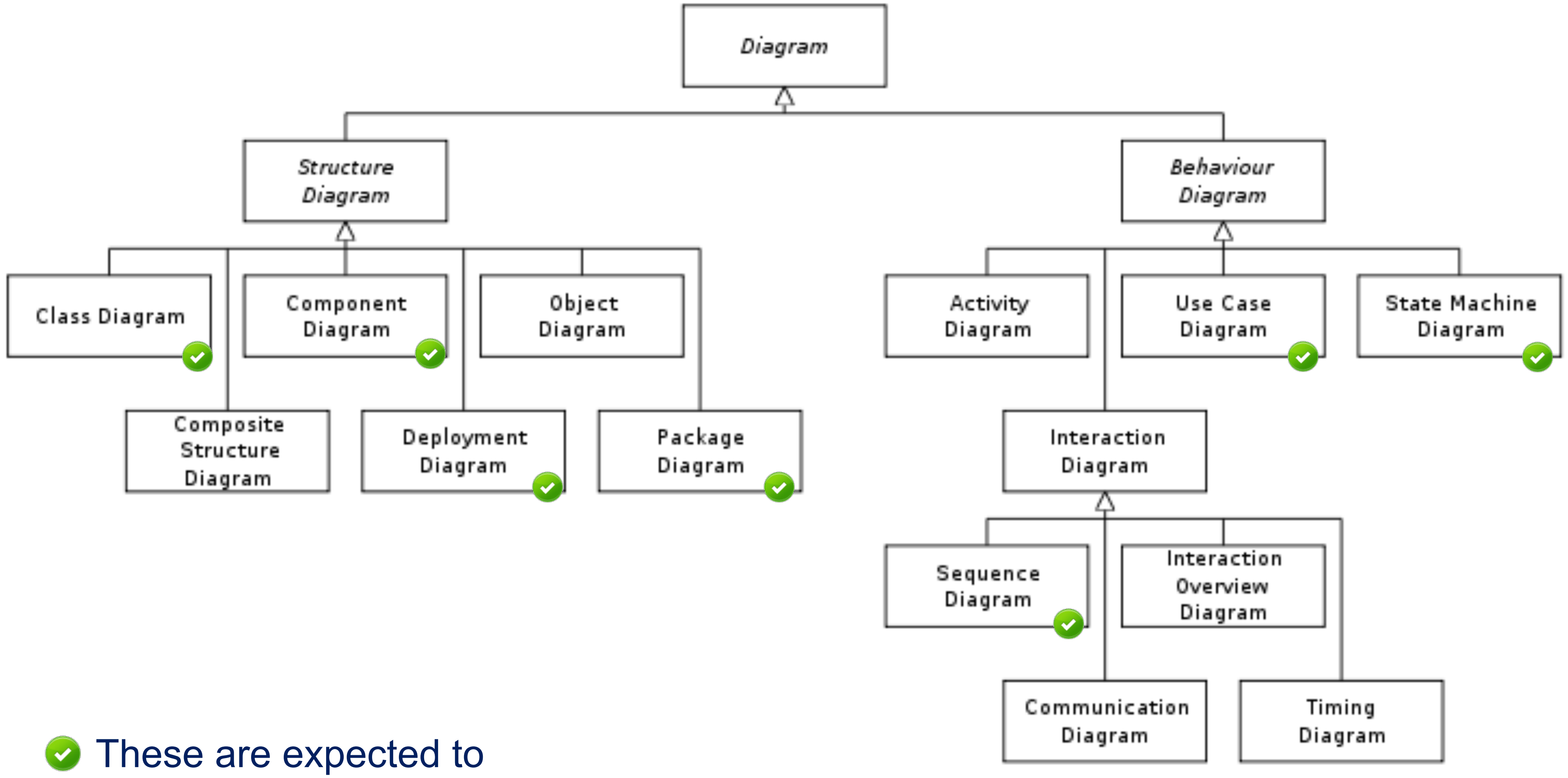


Note: Rectangle orange boxes denote artifacts (products),; whereas oval-shaped boxes denote processes (activities)

OO modeling in software engineering process II

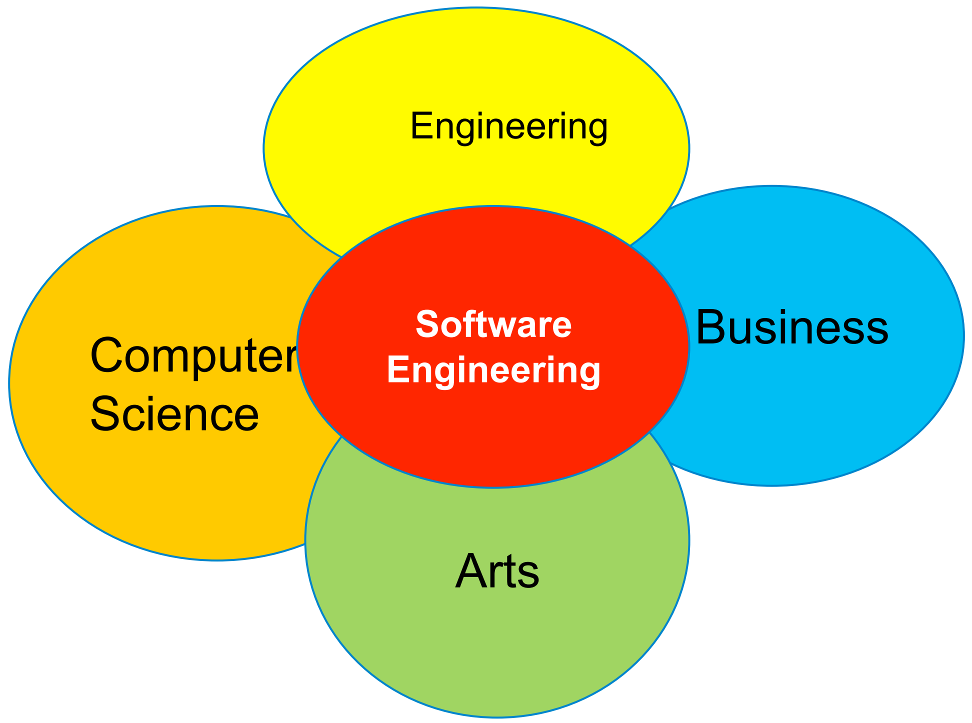


UML 2.0 - 13 Types of Diagrams



✓ These are expected to cover in this course

Software Engineering has intersection with other disciplines



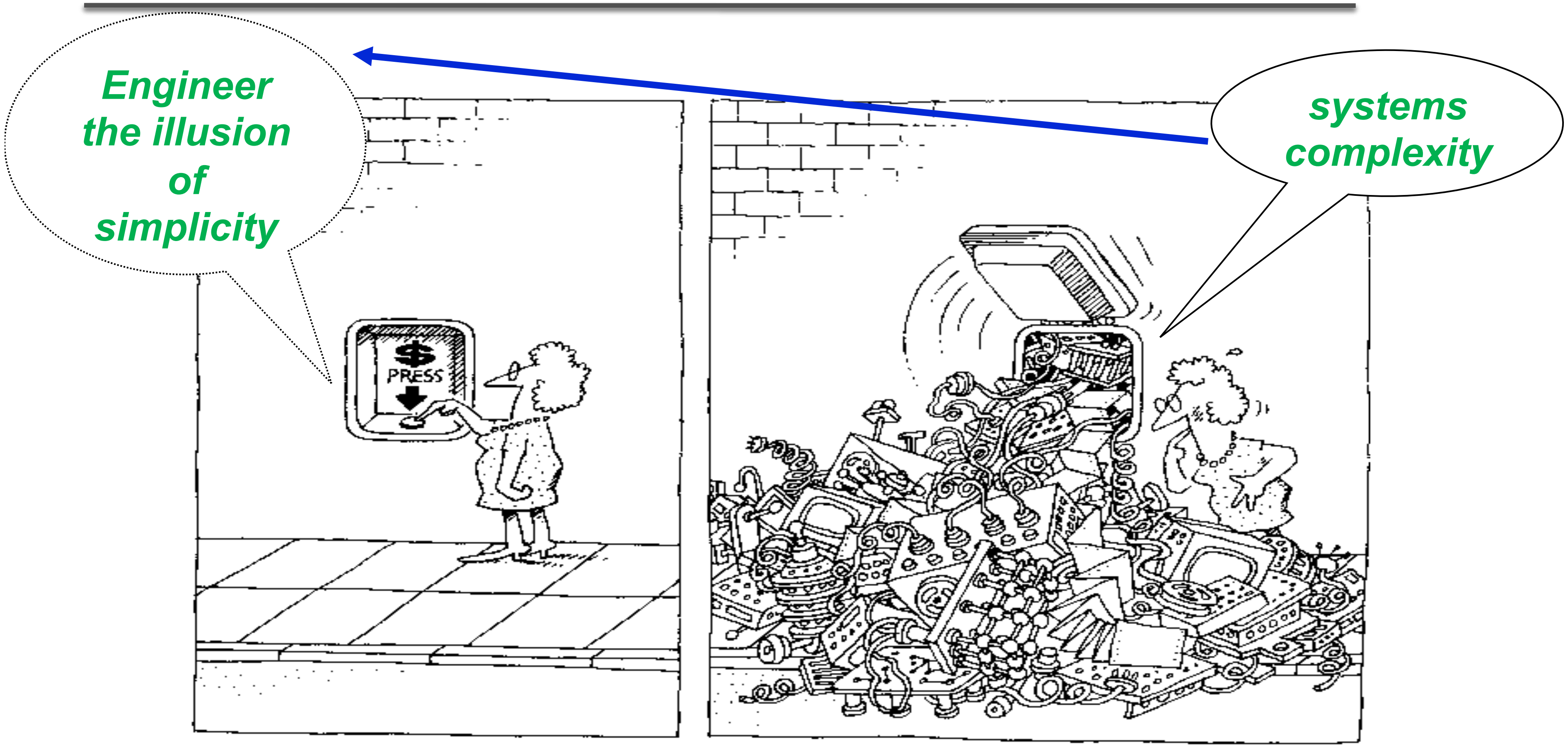
Software Engineering vs. Programming

Software engineering	Programming
Focuses on the entire system	Focuses on a part of the system
Interaction among elements	Implementations of elements
Structural properties	Computational properties
Usually static	Usually dynamic
System level performance	Algorithm level performance
Outside module boundary	Inside module boundary

Complexity and software

- Complexity of software is an essential property and derives from 4 elements
    1. The complexity of problem domain
    2. The difficulty of managing the software development process
    3. The constantly changing requirements of software
    4. The problems of characterizing the behavior of discrete systems
- We can’t quantify/measure complexity
- The complexity of software system often exceeds the human intellectual capacity
- There are fundamental limiting factors of human cognition
- One of the tasks in software engineering is to tackle this complexity and **engineer the illusion of simplicity** (*analysis and design*) for the system implementation (*programming*) !!!

Engineering the illusion of simplicity

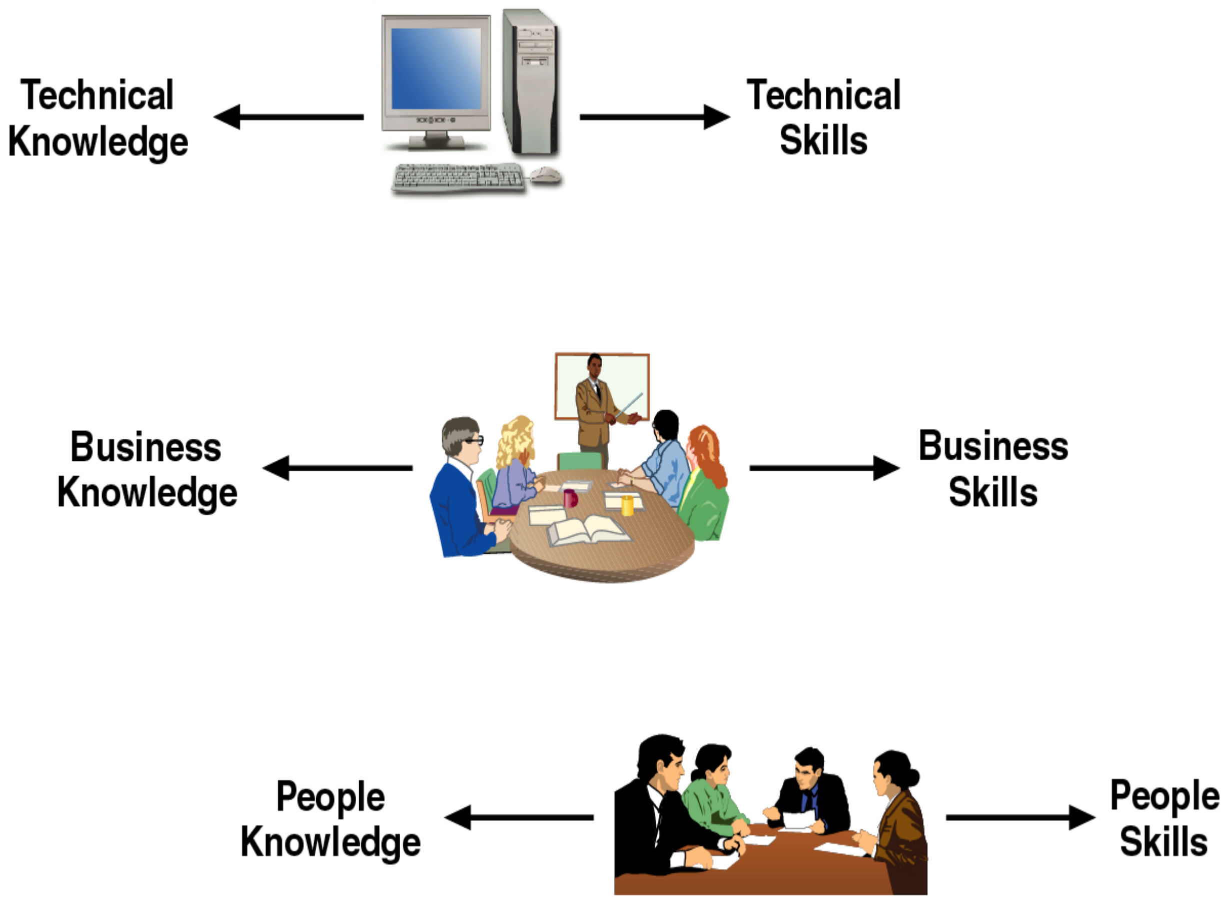


The task of the software development team is to engineer the illusion of simplicity.

Source: Booch, G.: Object Oriented Analysis and Design with Applications, Addison-Wesley, 2003, Chapter 1,

Required Skills of Software Engineers

Knowledge and Skills Required of a Systems Analyst



Understand the problem (what): An important activity

- ✧ *Who has a stake in the solution to the problem?* That is, who are the stakeholders?
- ✧ *What are the unknowns?* What data, functions, and features are required to properly solve the problem?
- ✧ *Can the problem be compartmentalized?* Is it possible to represent smaller problems that may be easier to understand?
- ✧ *Can the problem be represented graphically?* Can an analysis model be created?
- ✧ Is it the right software product that the client wants? How to confirm this?

Plan the solution (How)

- *Have you seen similar problems before?* Are there patterns that are recognizable in a potential solution? Is there existing software that implements the data, functions, and features that are required?
- *Has a similar problem been solved?* If so, are elements of the solution reusable?
- *Can subproblems be defined?* If so, are solutions readily apparent for the subproblems?
- *Can you represent a solution in a manner that leads to effective implementation?* Can a design model be created?
- Which techniques and tools should be adopted?
- Which strategies best suit for this project?

Carry out the plan (Implementation)

- *Does the solutions conform to the plan?* Is source code traceable to the design model?
- *Is each component part of the solution provably correct?* Has the design and code been reviewed, or better, have correctness proofs been applied to algorithm?
- Which languages should be used? Why?

Examine the result (Testing)

- ✧ *Is it possible to test each component part of the solution?* Has a reasonable testing strategy been implemented?
- ✧ *Does the solution produce results that conform to the data, functions, and features that are required?* Has the software been validated against all stakeholder requirements?

Professional and ethical responsibility

- Software engineers must behave in an honest and ethically responsible way
- Ethical behaviour is more than simply upholding the law
  - ***Confidentiality***: Engineers should normally respect the confidentiality of their employers or clients
  - ***Competence***: Engineers should not misrepresent their level of competence. Should not knowingly accept work beyond their competence
  - ***Intellectual Property rights****:* Engineers should be aware of local laws governing the use of intellectual property such as patents, copyright
  - ***Computer misuse***: Should not misuse their technical skills (virus)

Code of Ethics - principles from IEEE/ACM

- ✧ **Public**
  - Software engineers shall act consistently with the public interest.
- ✧ **Client and Employer**
  - Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
- ✧ **Product**
  - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
- ✧ **Management**
  - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.

Code of Ethics – principles (continued)

- **Judgment**
  - Software engineers shall maintain integrity and independence in their professional judgment.
- **Profession**
  - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
- **Colleagues**
  - Software engineers shall be fair to and supportive of their colleagues.
- **Self**
  - Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

Terminologies in software engineering

- Software development process
- Software process
- Software life cycle process
- Software life cycle model
- Software development method
- Software development methodology
- Software development life cycle
- Software development model

All these can be used interchangeably

Key points

- Software engineering is an engineering discipline that is concerned with all aspects of software production
- Essential software product attributes are maintainability, dependability and security, efficiency and acceptability
- The high-level activities of specification, development, validation and evolution are part of all software processes
- The fundamental notions of software engineering are universally applicable to all types of system development
- There are many different types of system and each requires appropriate software engineering tools and techniques for their development
- Software engineers must adhere with the ethical aspects of the profession