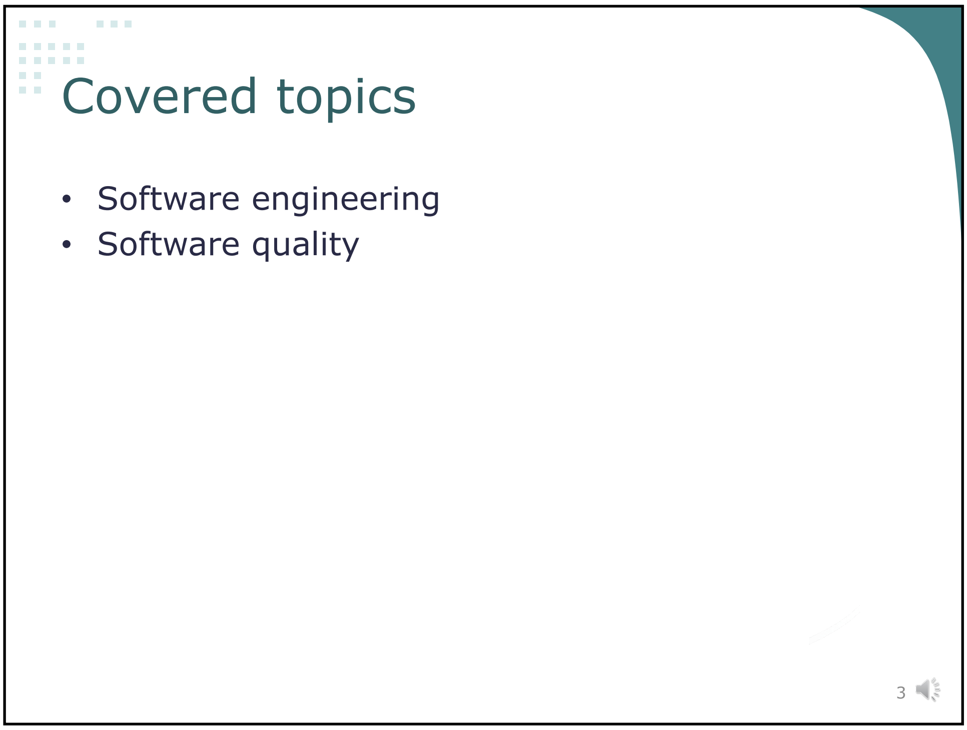
Structure Programming
Overview of Software Engineering

Dr. Nguyen Thanh Hung
Software Engineering Department
School of ICT

@2017-2018

1

2

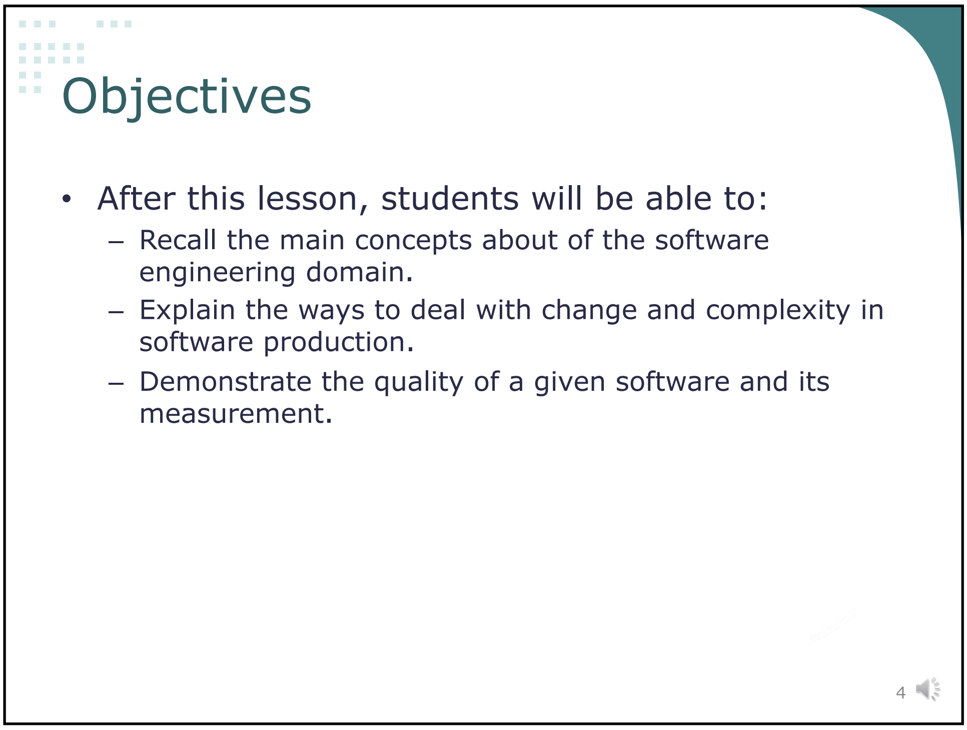


Covered topics

- Software engineering
- Software quality

3

3

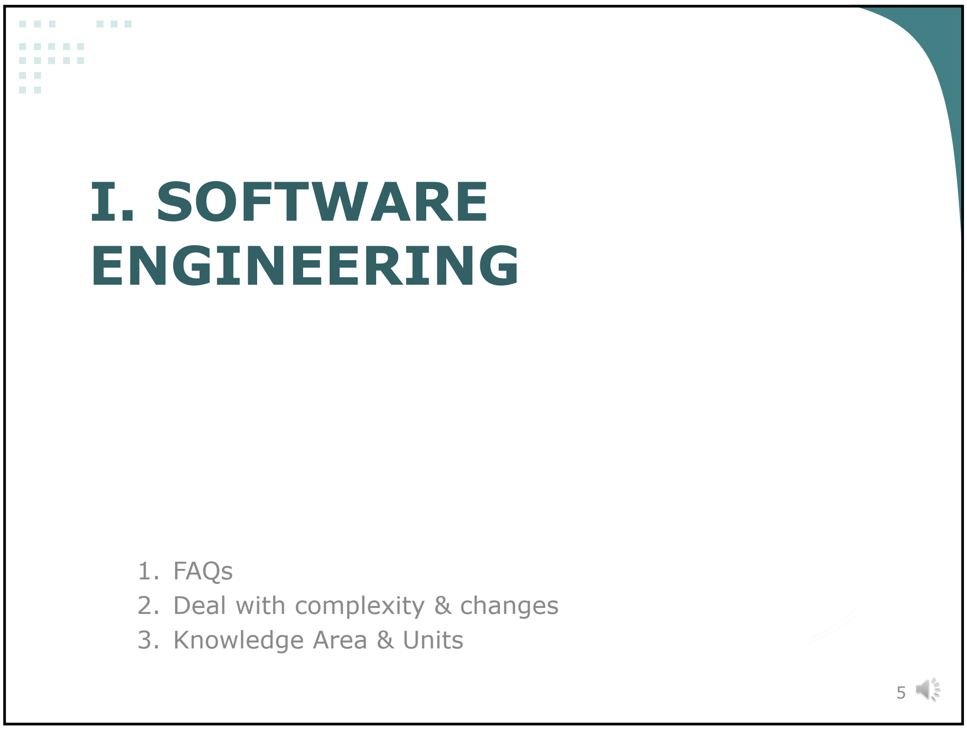


Objectives

- After this lesson, students will be able to:
  - Recall the main concepts about of the software engineering domain.
  - Explain the ways to deal with change and complexity in software production.
  - Demonstrate the quality of a given software and its measurement.

4

4

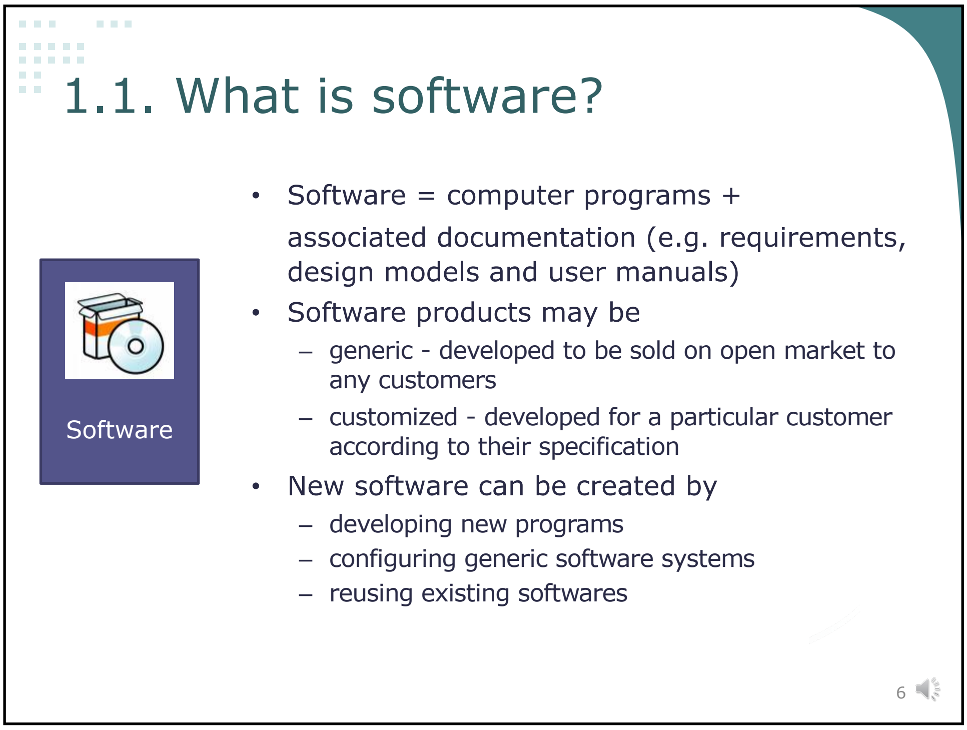


I. SOFTWARE ENGINEERING

1. FAQs
2. Deal with complexity & changes
3. Knowledge Area & Units

5

5



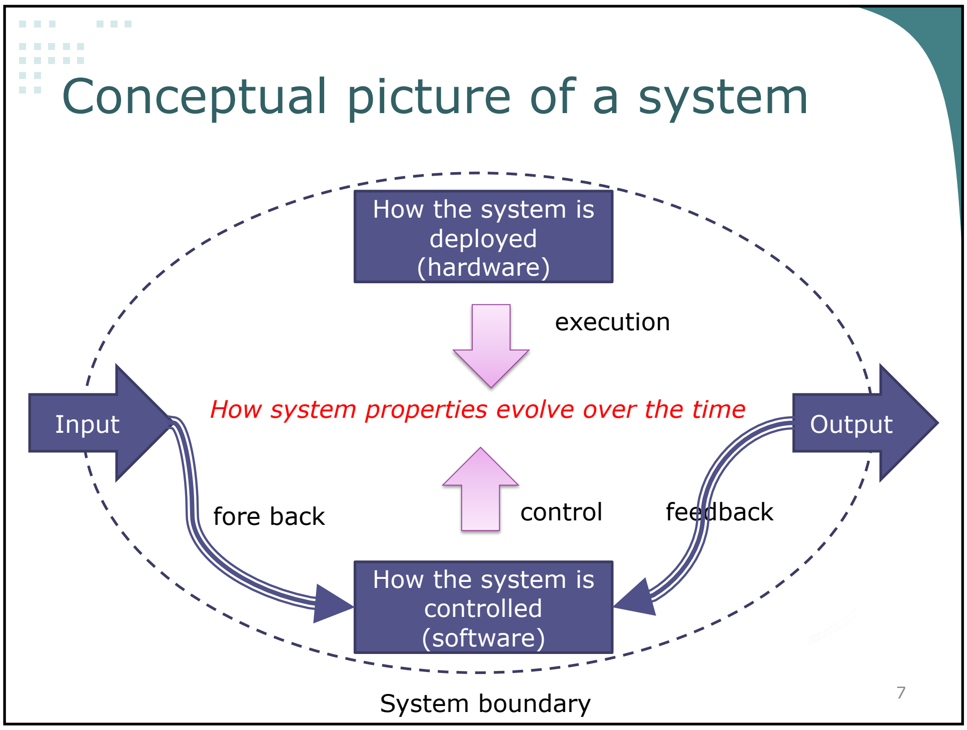
1.1. What is software?

Software

- Software = computer programs + associated documentation (e.g. requirements, design models and user manuals)
- Software products may be
  - generic - developed to be sold on open market to any customers
  - customized - developed for a particular customer according to their specification
- New software can be created by
  - developing new programs
  - configuring generic software systems
  - reusing existing softwares

6

6



Conceptual picture of a system

How the system is deployed (hardware)

execution

How system properties evolve over the time

Input

Output

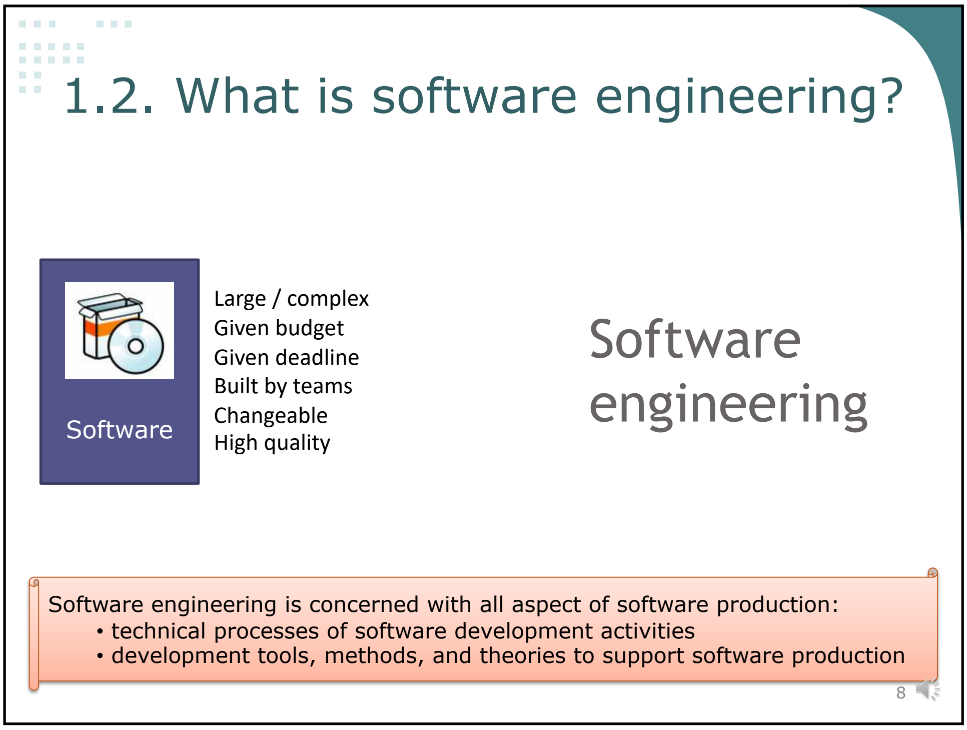
fore back

control

feedback

How the system is controlled (software)

System boundary



1.2. What is software engineering?

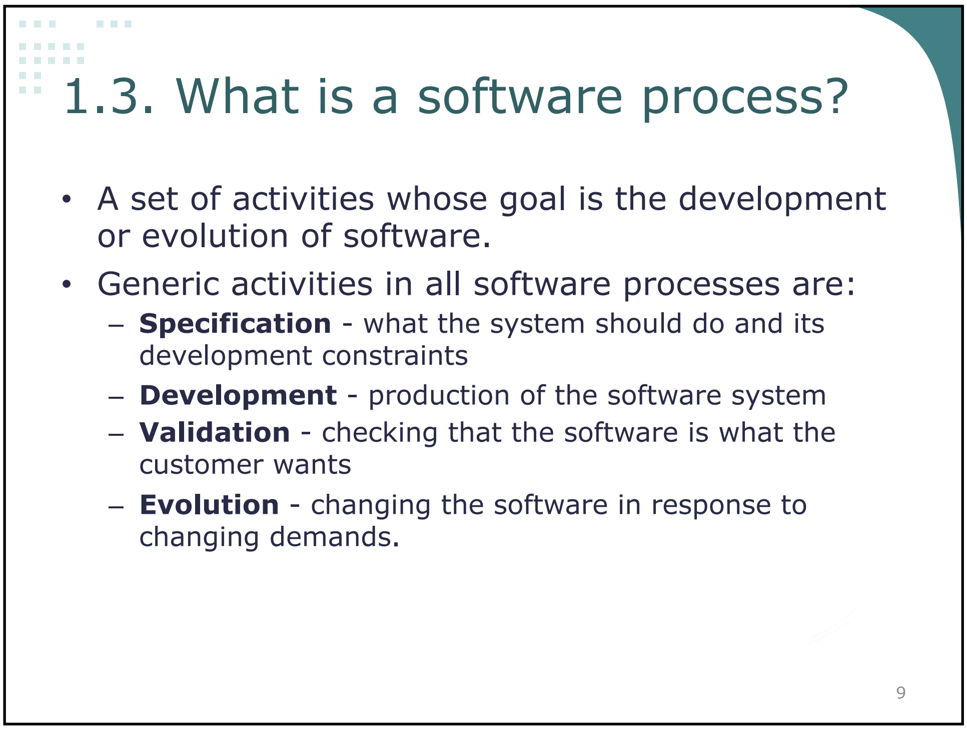
Software

Large / complex
Given budget
Given deadline
Built by teams
Changeable
High quality

Software engineering

Software engineering is concerned with all aspect of software production:

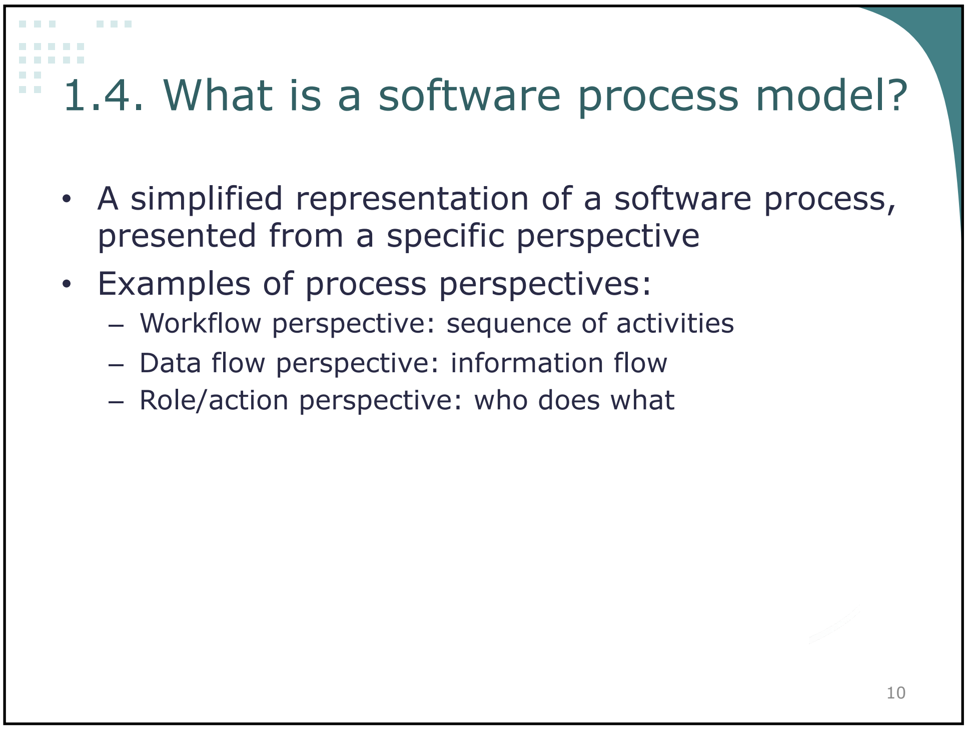
- technical processes of software development activities
- development tools, methods, and theories to support software production



1.3. 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.

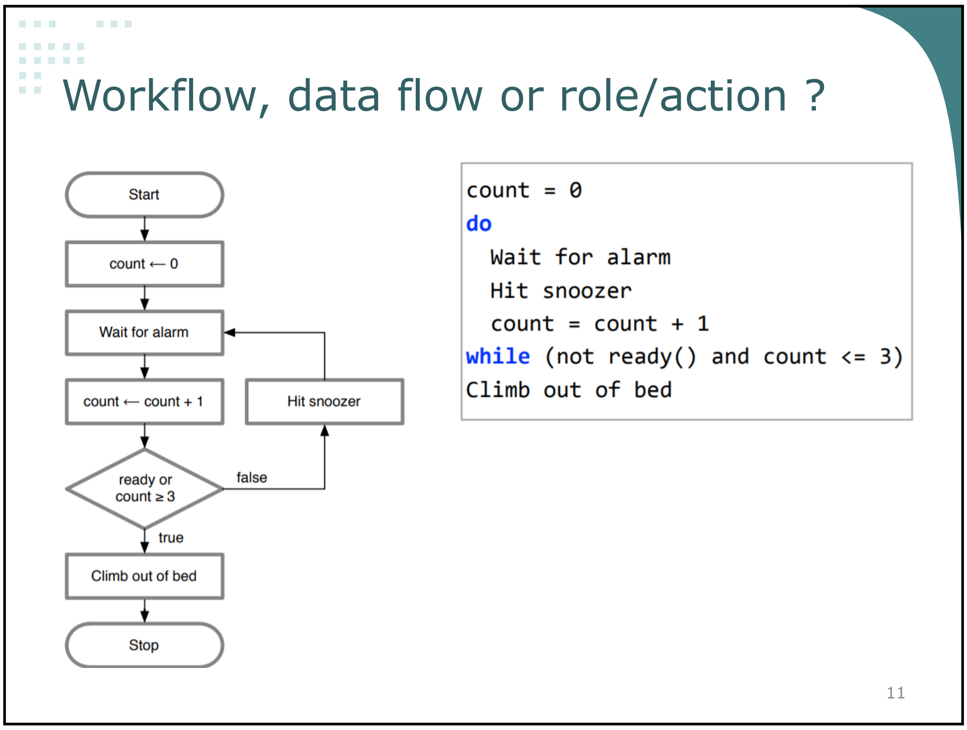
9



1.4. What is a software process model?

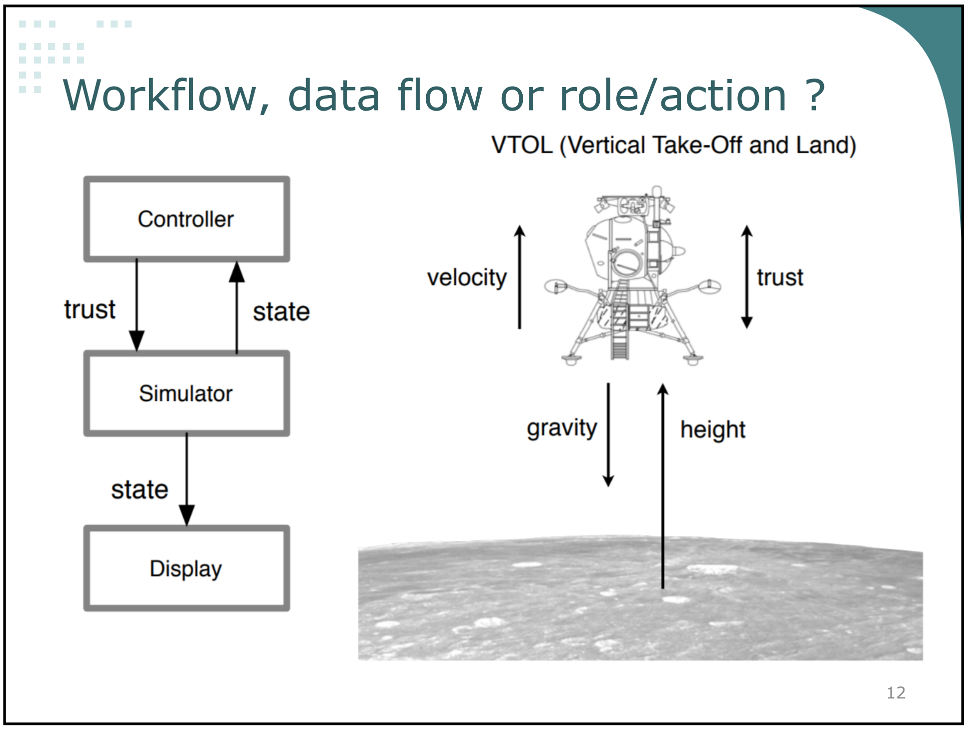
- A simplified representation of a software process, presented from a specific perspective
- Examples of process perspectives:
  - Workflow perspective: sequence of activities
  - Data flow perspective: information flow
  - Role/action perspective: who does what

10



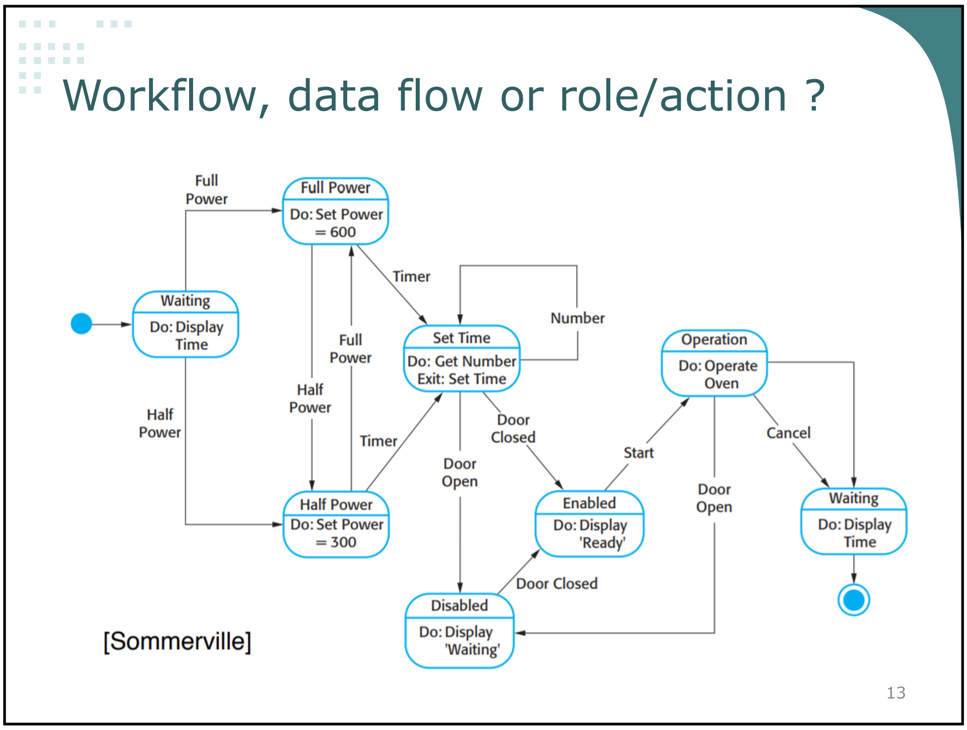
Workflow, data flow or role/action ?

```
count = 0
do
  Wait for alarm
  Hit snoozer
  count = count + 1
while (not ready() and count <= 3)
Climb out of bed
```



Workflow, data flow or role/action ?

VTOL (Vertical Take-Off and Land)

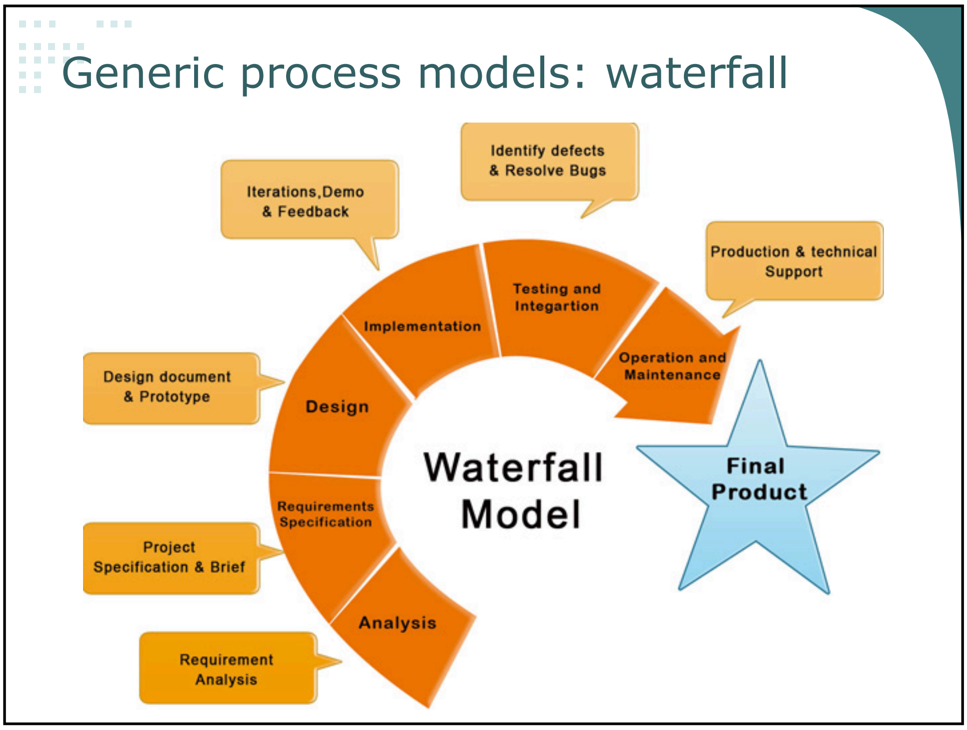


Workflow, data flow or role/action ?

[Sommerville]

13

13

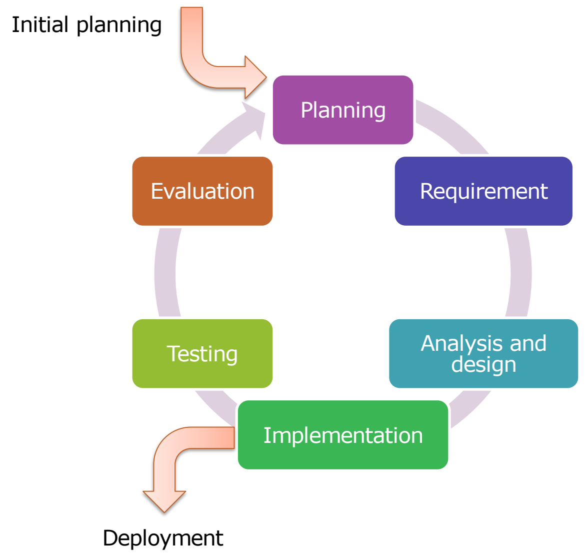


Generic process models: waterfall

Identify defects & Resolve Bugs
Iterations,Demo & Feedback
Production & technical Support
Testing and Integartion
Implementation
Operation and Maintenance
Design document & Prototype
Design
Waterfall Model
Final Product
Requirements Specification
Project Specification & Brief
Analysis
Requirement Analysis

14

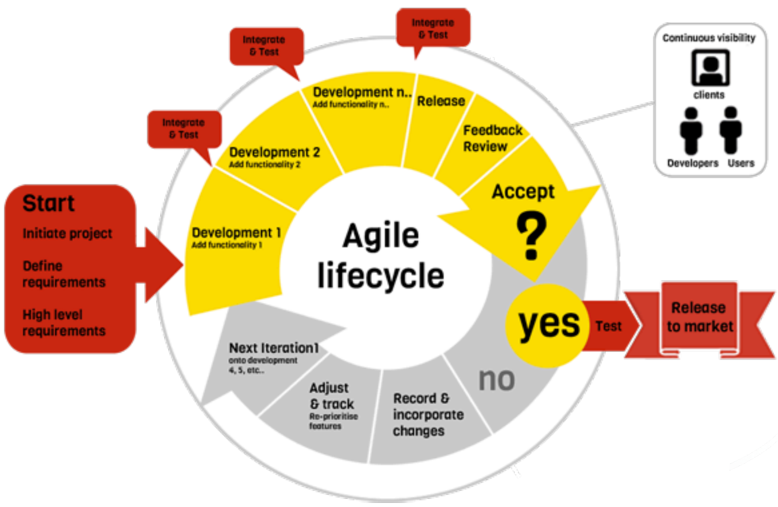
Generic process models: iterative



15

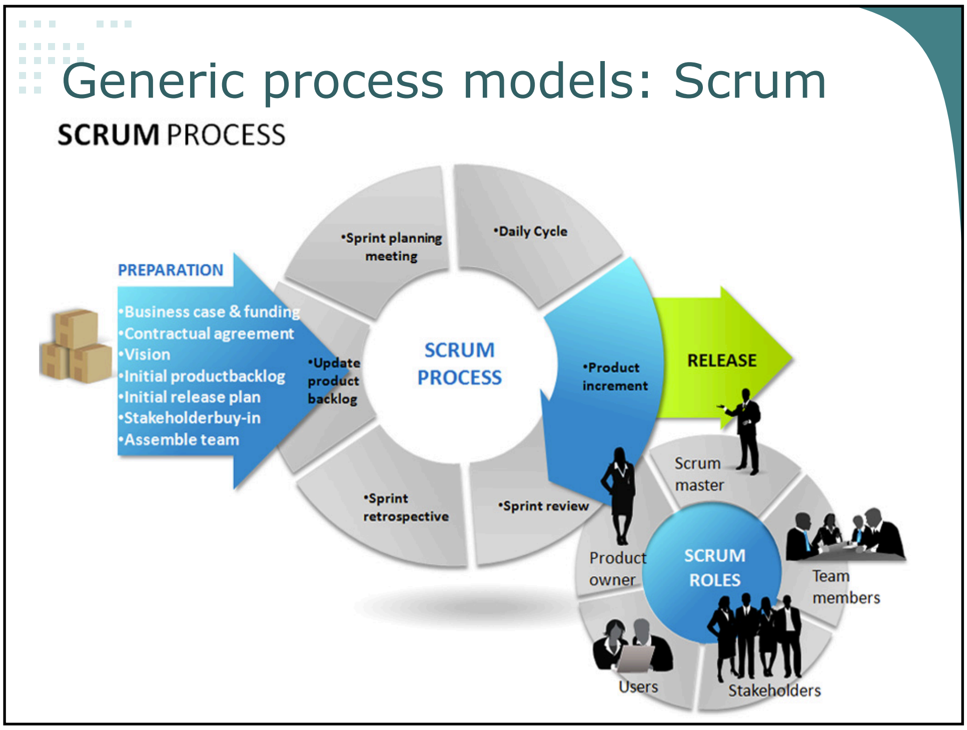
15

Generic process models: Agile



16

16



Generic process models: Scrum

SCRUM PROCESS

PREPARATION

- •Business case & funding
- •Contractual agreement
- •Vision
- •Initial productbacklog
- •Initial release plan
- •Stakeholderbuy-in
- •Assemble team

SCRUM PROCESS

•Sprint planning meeting

•Daily Cycle

•Product increment

•Sprint review

•Sprint retrospective

•Update product backlog

RELEASE

SCRUM ROLES

Scrum master

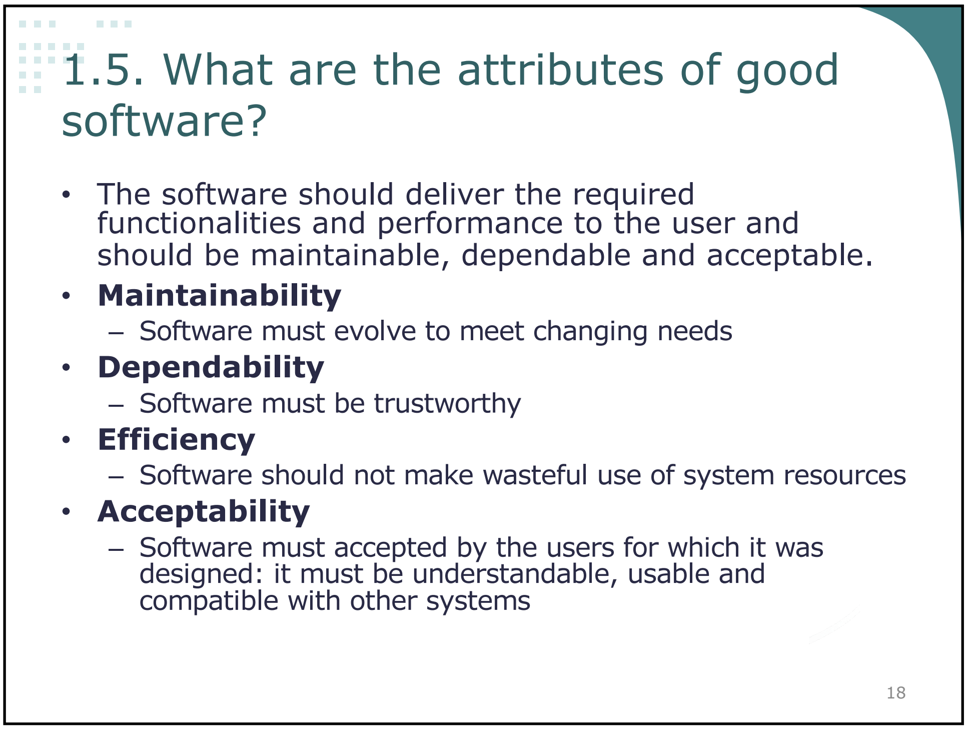
Team members

Stakeholders

Users

Product owner

17

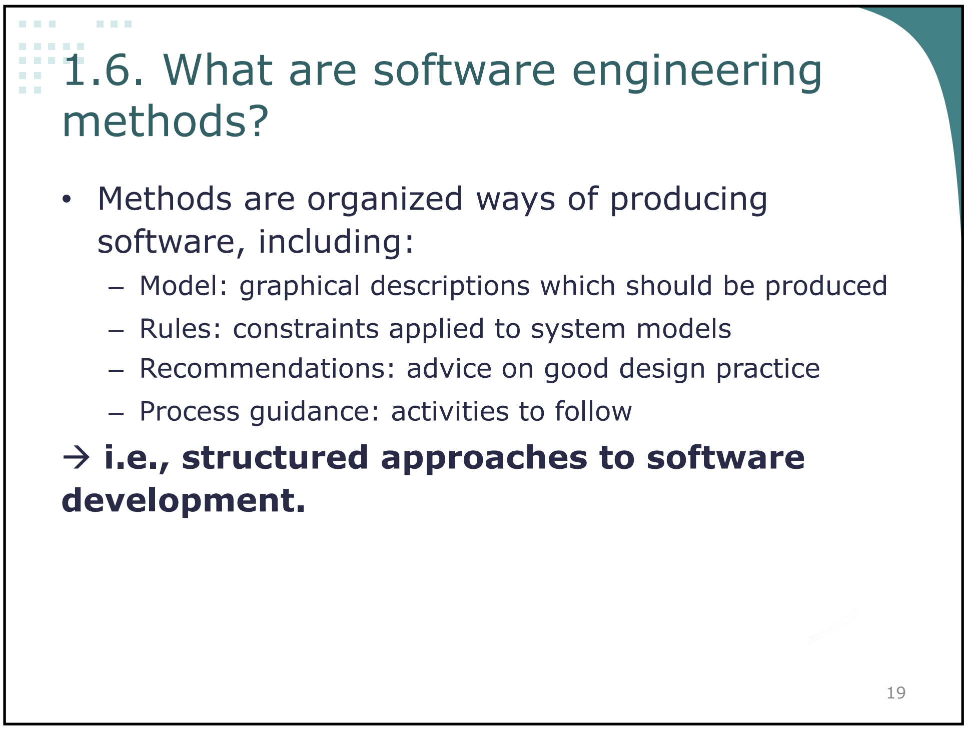


1.5. What are the attributes of good software?

- The software should deliver the required functionalities 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: it must be understandable, usable and compatible with other systems

18

18



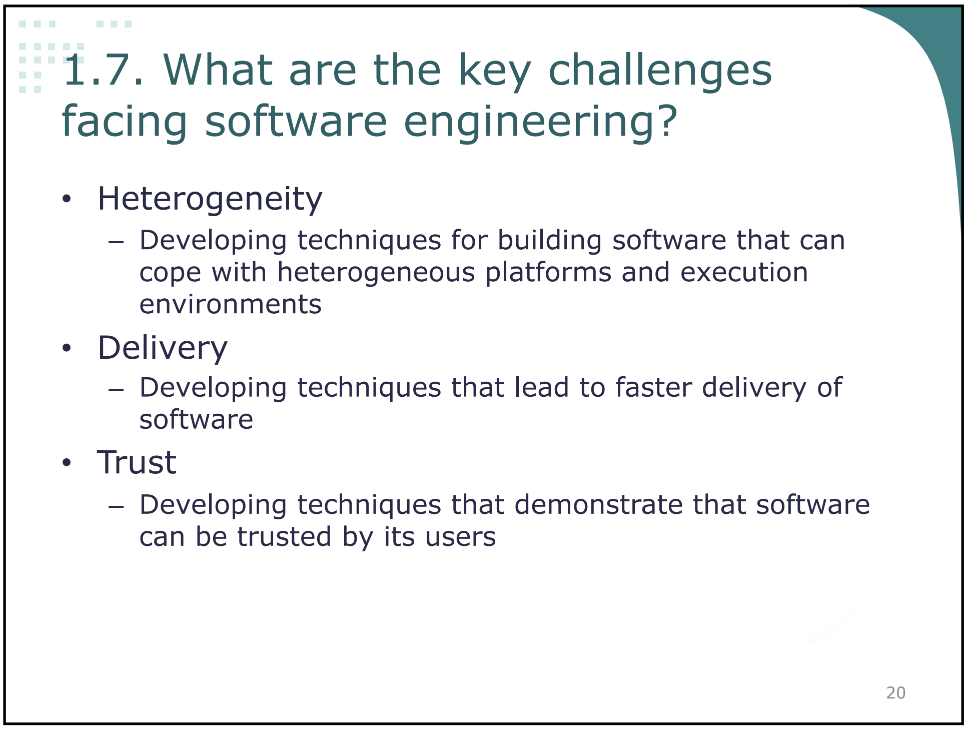
1.6. What are software engineering methods?

- Methods are organized ways of producing software, including:
  - Model: graphical descriptions which should be produced
  - Rules: constraints applied to system models
  - Recommendations: advice on good design practice
  - Process guidance: activities to follow

→ **i.e., structured approaches to software development.**

19

19

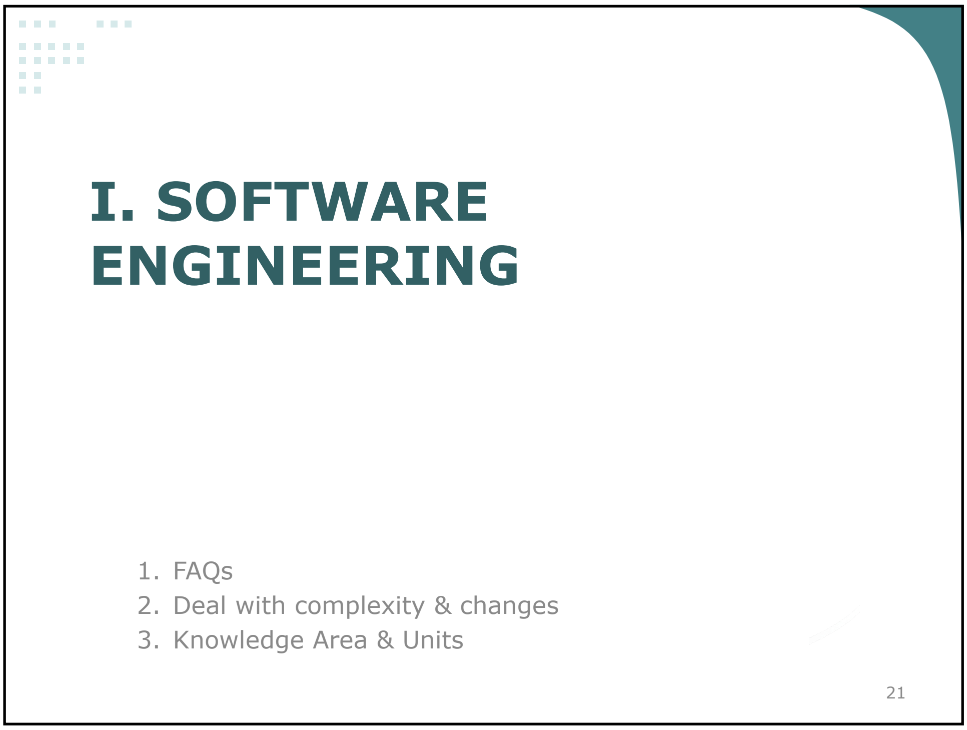


1.7. What are the key challenges facing software engineering?

- 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

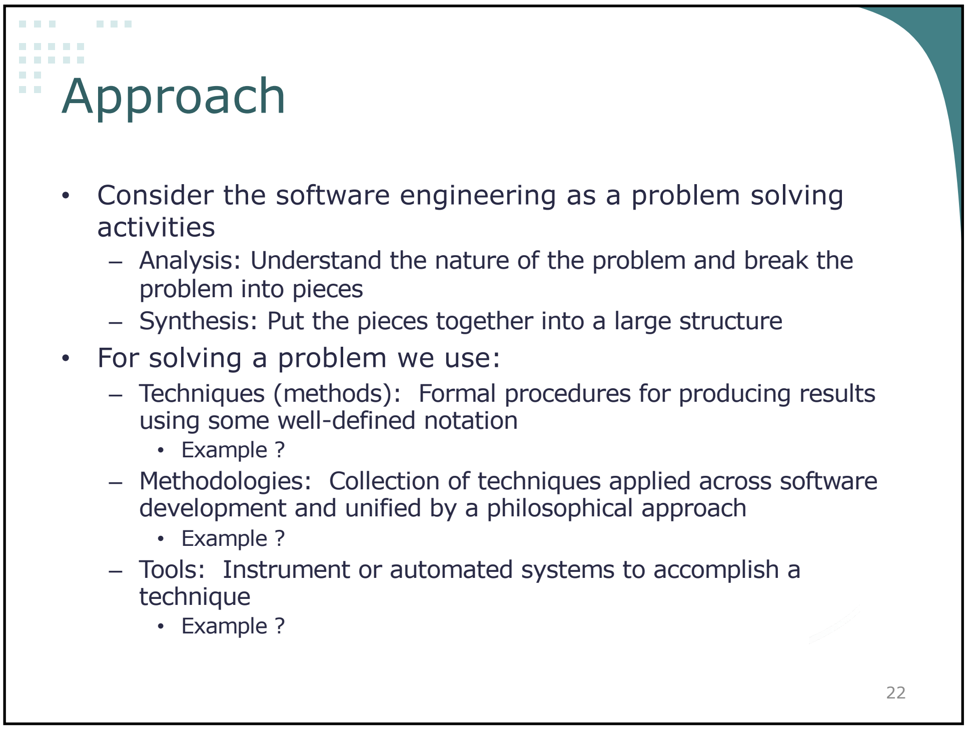
20

20



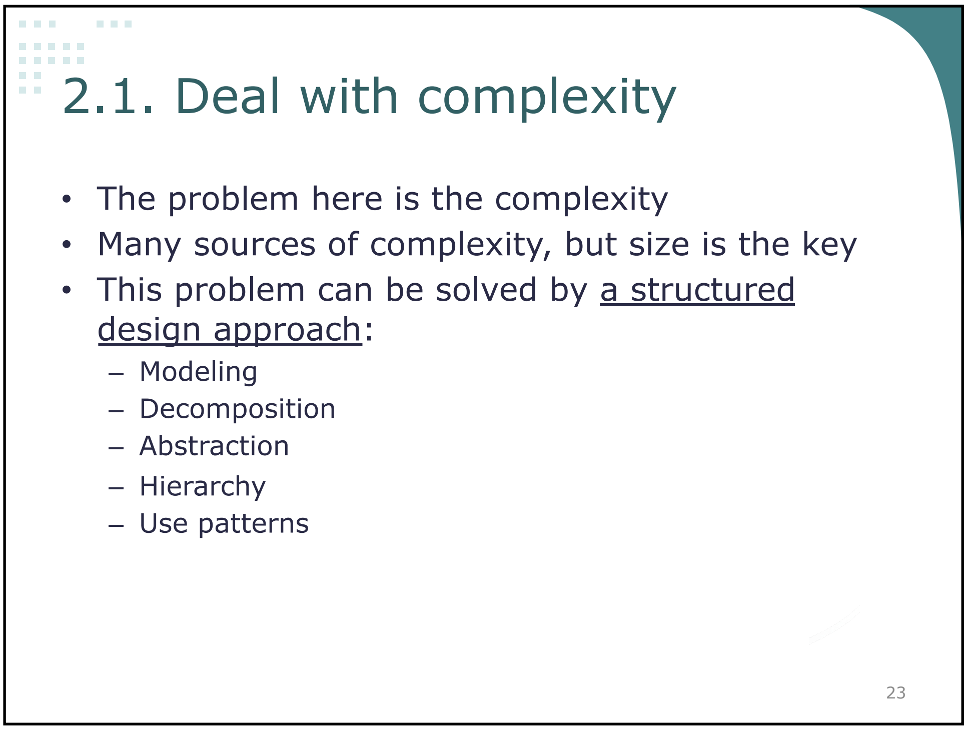
I. SOFTWARE ENGINEERING

1. FAQs
2. Deal with complexity & changes
3. Knowledge Area & Units



Approach

- Consider the software engineering as a problem solving activities
  - Analysis: Understand the nature of the problem and break the problem into pieces
  - Synthesis: Put the pieces together into a large structure
- For solving a problem we use:
  - Techniques (methods): Formal procedures for producing results using some well-defined notation
    - Example ?
  - Methodologies: Collection of techniques applied across software development and unified by a philosophical approach
    - Example ?
  - Tools: Instrument or automated systems to accomplish a technique
    - Example ?

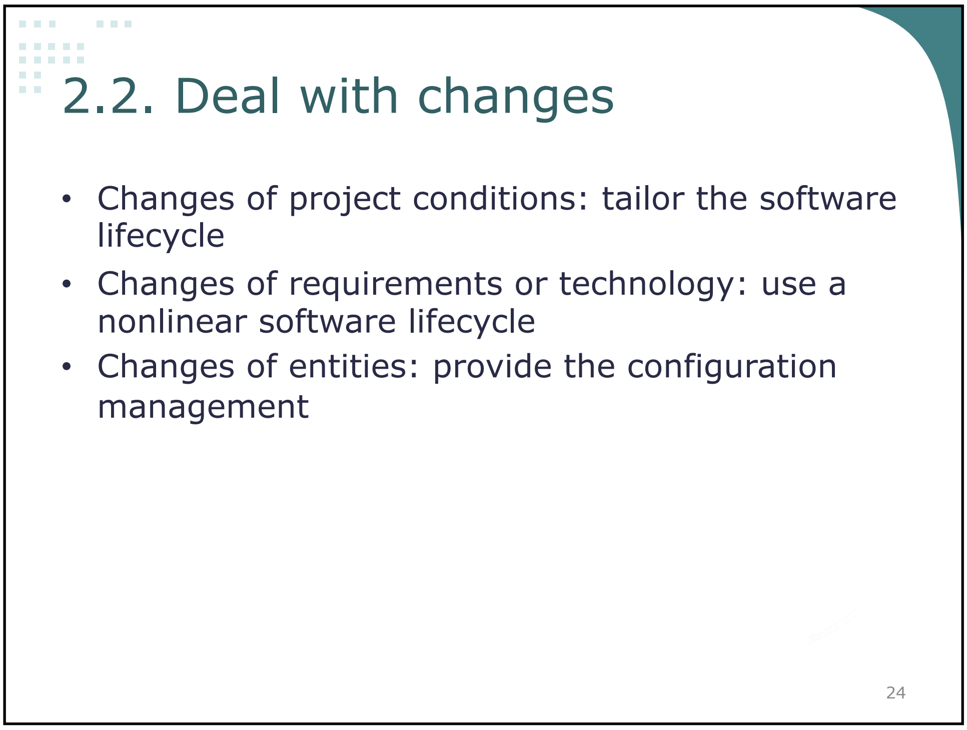


2.1. Deal with complexity

- The problem here is the complexity
- Many sources of complexity, but size is the key
- This problem can be solved by a structured design approach:
  - Modeling
  - Decomposition
  - Abstraction
  - Hierarchy
  - Use patterns

23

23

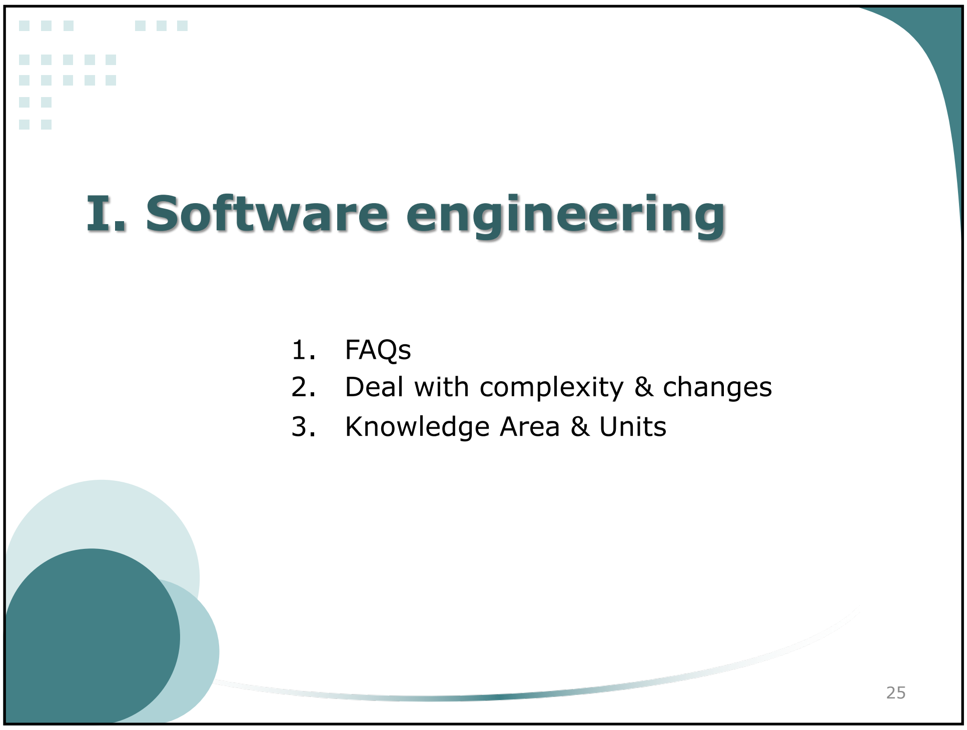


2.2. Deal with changes

- Changes of project conditions: tailor the software lifecycle
- Changes of requirements or technology: use a nonlinear software lifecycle
- Changes of entities: provide the configuration management

24

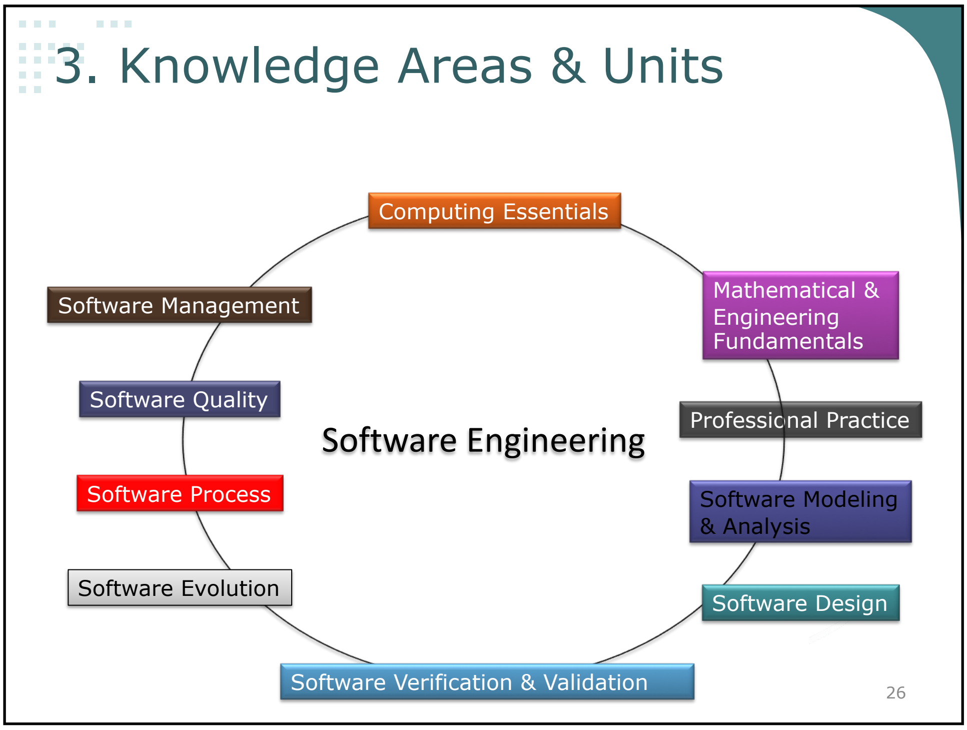
24



I. Software engineering

1. FAQs
2. Deal with complexity & changes
3. Knowledge Area & Units

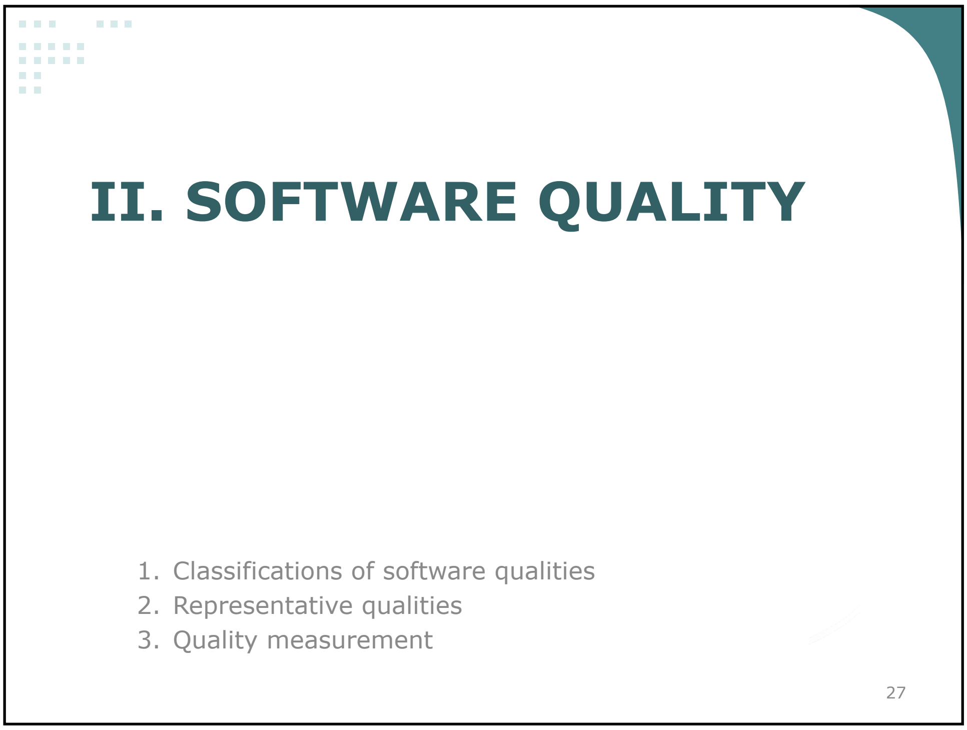
25



3. Knowledge Areas & Units

Software Engineering

26

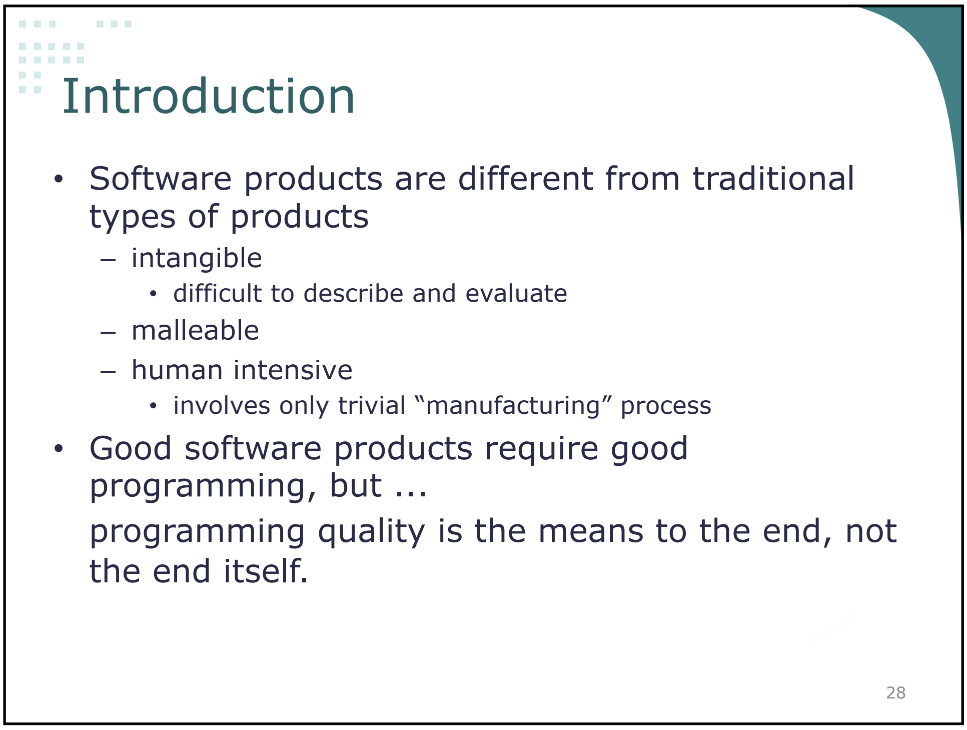


II. SOFTWARE QUALITY

1. Classifications of software qualities
2. Representative qualities
3. Quality measurement

27

27



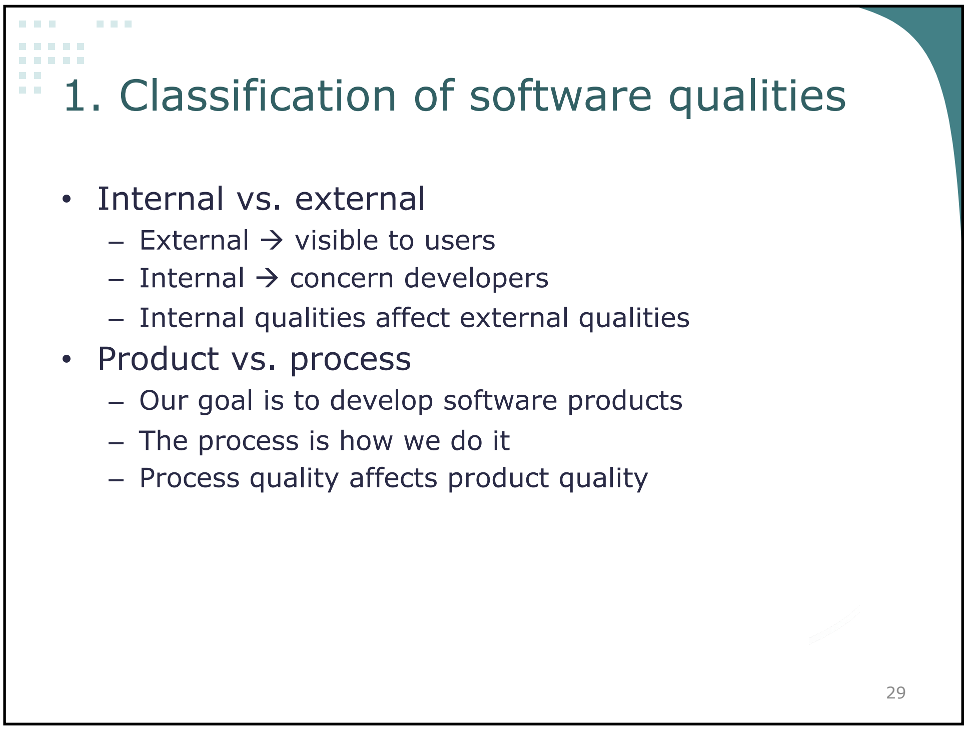
Introduction

- Software products are different from traditional types of products
  - intangible
    - difficult to describe and evaluate
  - malleable
  - human intensive
    - involves only trivial "manufacturing" process
- Good software products require good programming, but ...

  programming quality is the means to the end, not the end itself.

28

28

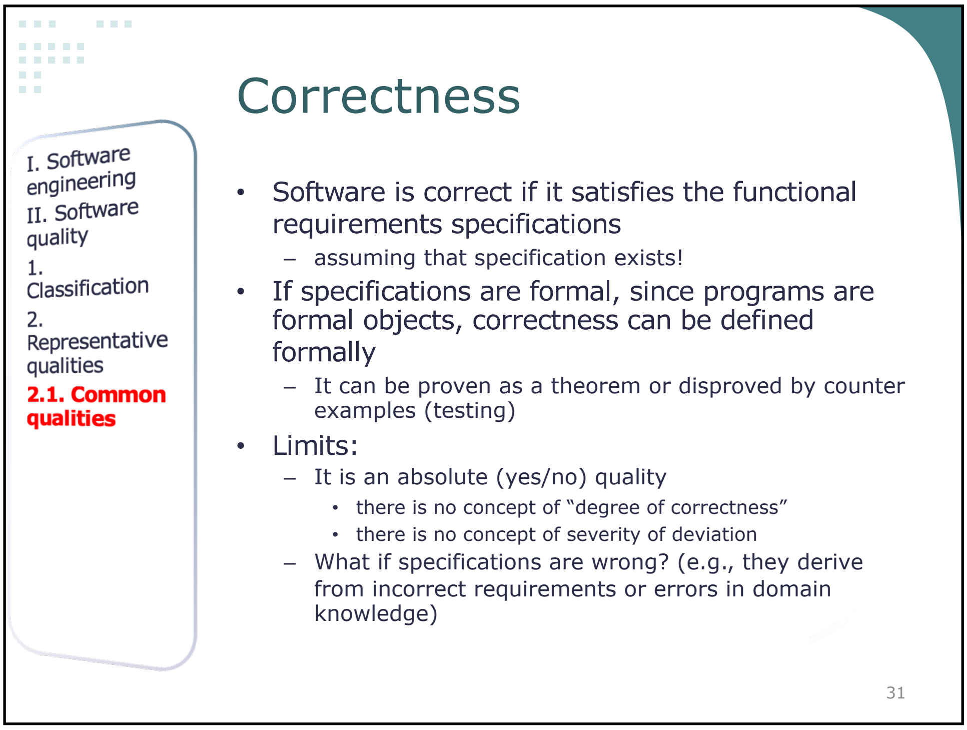


1. Classification of software qualities

- Internal vs. external
  - External → visible to users
  - Internal → concern developers
  - Internal qualities affect external qualities
- Product vs. process
  - Our goal is to develop software products
  - The process is how we do it
  - Process quality affects product quality

29

29



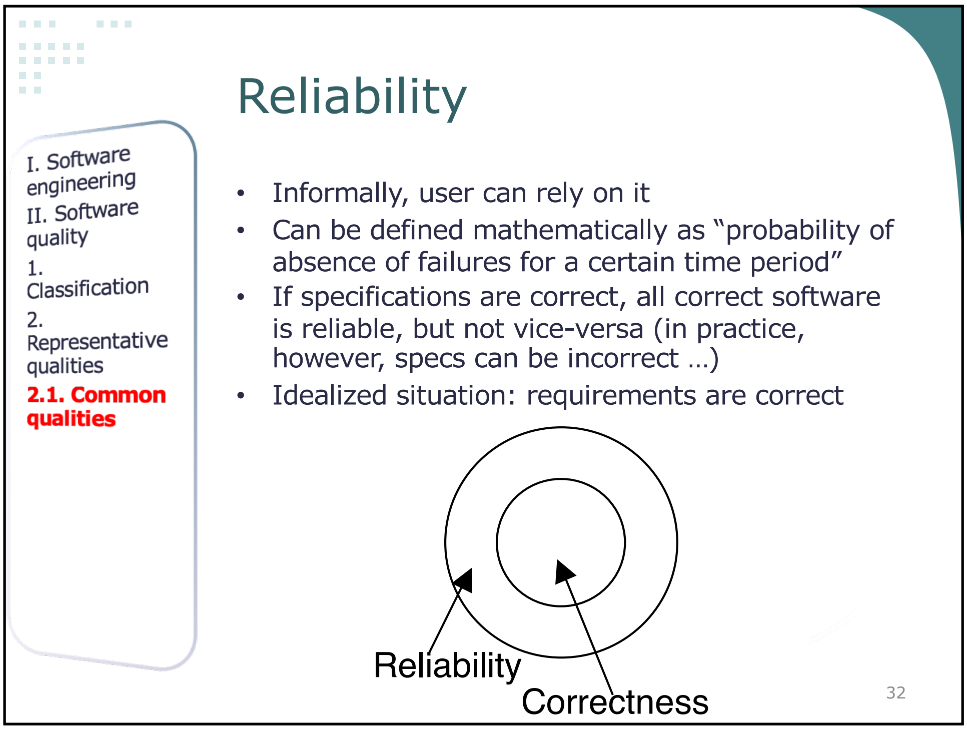
I. Software engineering
II. Software quality
1. Classification
2. Representative qualities
2.1. Common qualities

Correctness

- Software is correct if it satisfies the functional requirements specifications
  - assuming that specification exists!
- If specifications are formal, since programs are formal objects, correctness can be defined formally
  - It can be proven as a theorem or disproved by counter examples (testing)
- Limits:
  - It is an absolute (yes/no) quality
    - there is no concept of "degree of correctness"
    - there is no concept of severity of deviation
  - What if specifications are wrong? (e.g., they derive from incorrect requirements or errors in domain knowledge)

31

31



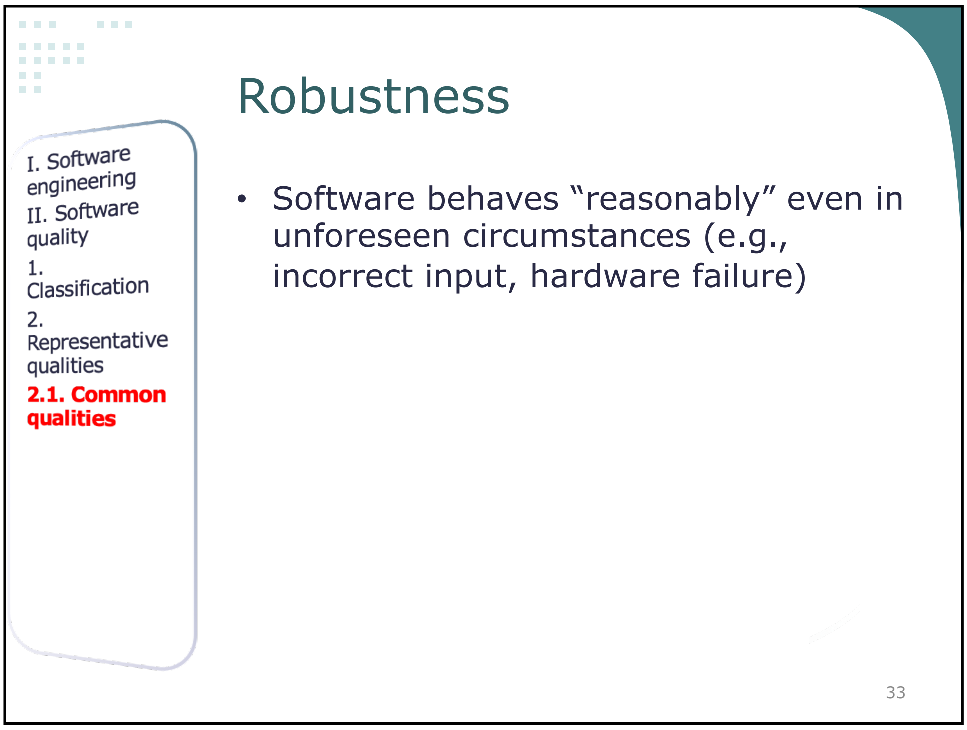
I. Software engineering
II. Software quality
1. Classification
2. Representative qualities
2.1. Common qualities

Reliability

- Informally, user can rely on it
- Can be defined mathematically as "probability of absence of failures for a certain time period"
- If specifications are correct, all correct software is reliable, but not vice-versa (in practice, however, specs can be incorrect …)
- Idealized situation: requirements are correct

32

32



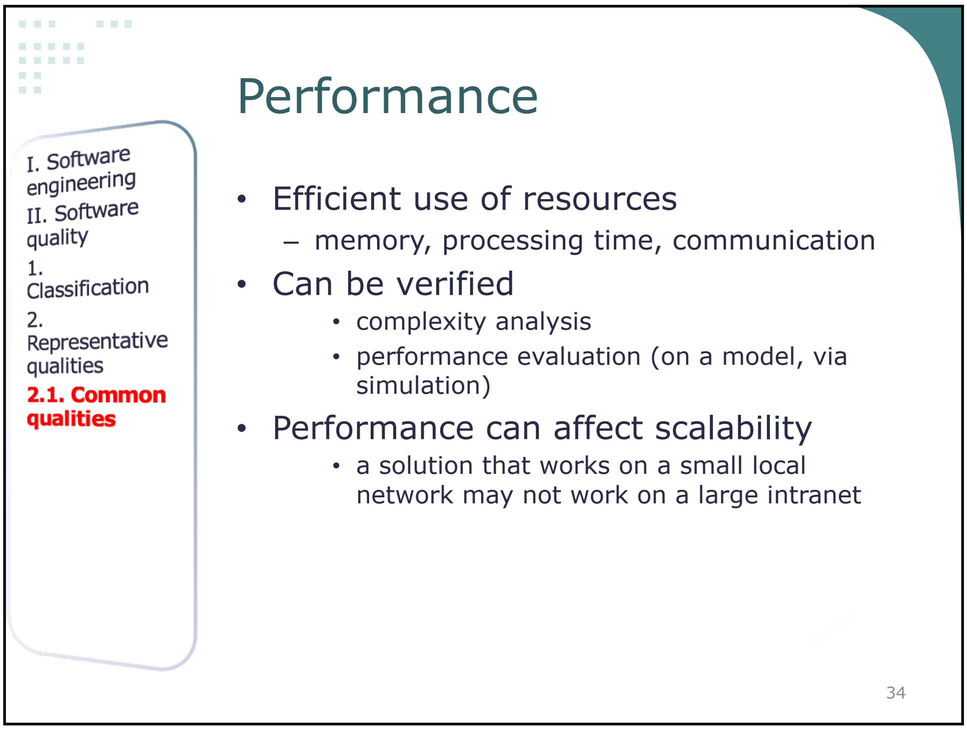
I. Software engineering
II. Software quality
1. Classification
2. Representative qualities
2.1. Common qualities

Robustness

- Software behaves "reasonably" even in unforeseen circumstances (e.g., incorrect input, hardware failure)

33

33



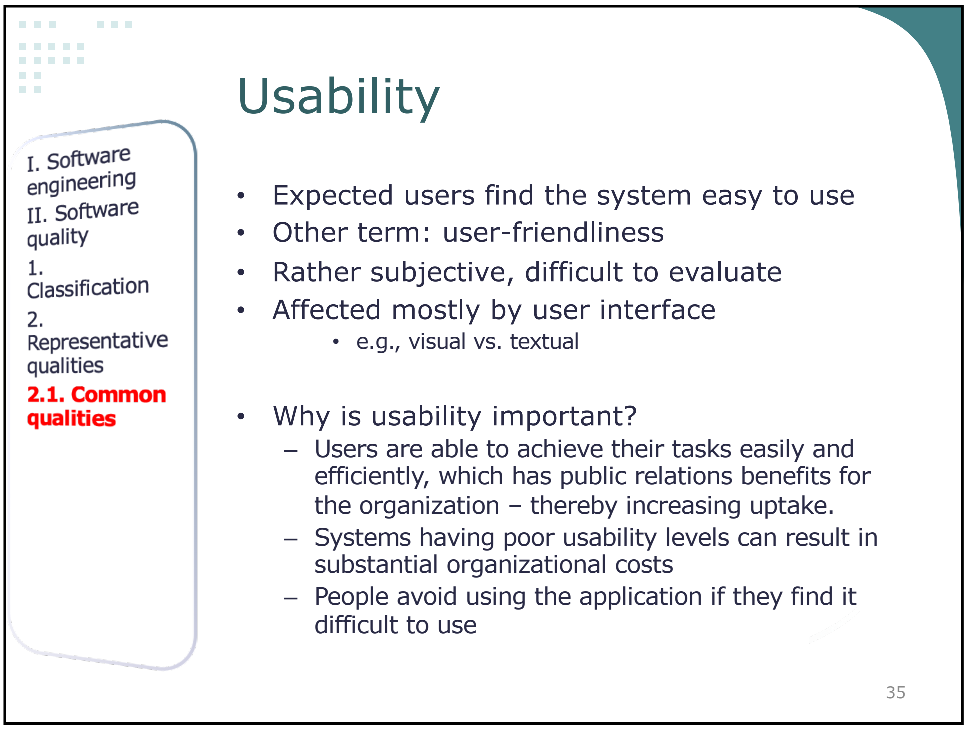
I. Software engineering
II. Software quality
1. Classification
2. Representative qualities
2.1. Common qualities

Performance

- Efficient use of resources
  - memory, processing time, communication
- Can be verified
  - complexity analysis
  - performance evaluation (on a model, via simulation)
- Performance can affect scalability
  - a solution that works on a small local network may not work on a large intranet

34

34



I. Software engineering
II. Software quality
1. Classification
2. Representative qualities
2.1. Common qualities

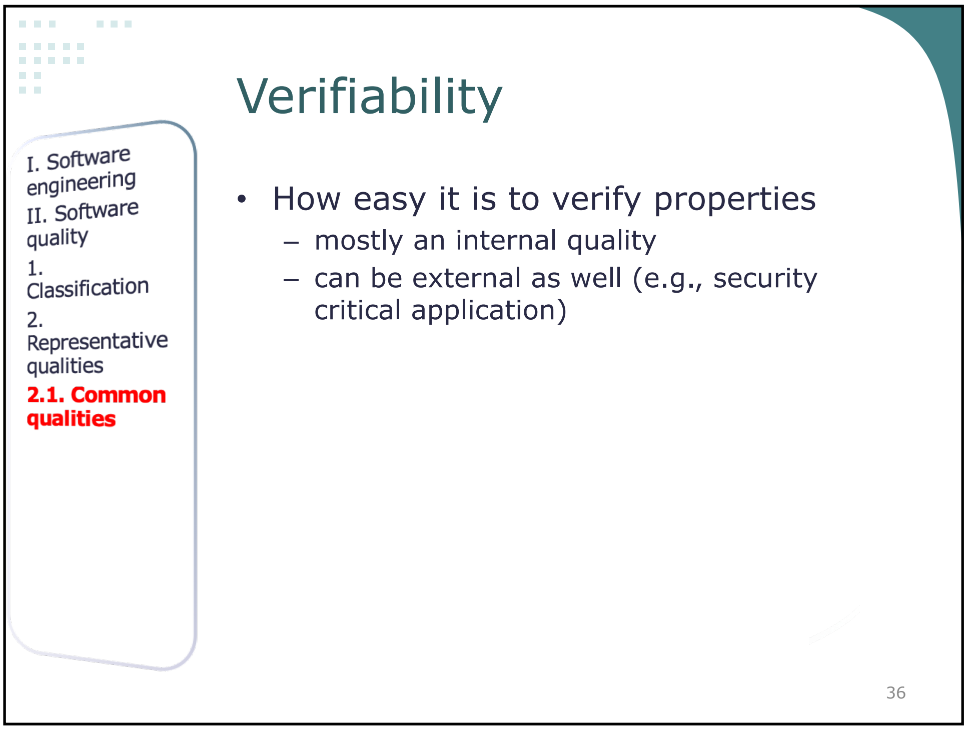
Usability

- Expected users find the system easy to use
- Other term: user-friendliness
- Rather subjective, difficult to evaluate
- Affected mostly by user interface
  - e.g., visual vs. textual

- Why is usability important?
  - Users are able to achieve their tasks easily and efficiently, which has public relations benefits for the organization – thereby increasing uptake.
  - Systems having poor usability levels can result in substantial organizational costs
  - People avoid using the application if they find it difficult to use

35

35



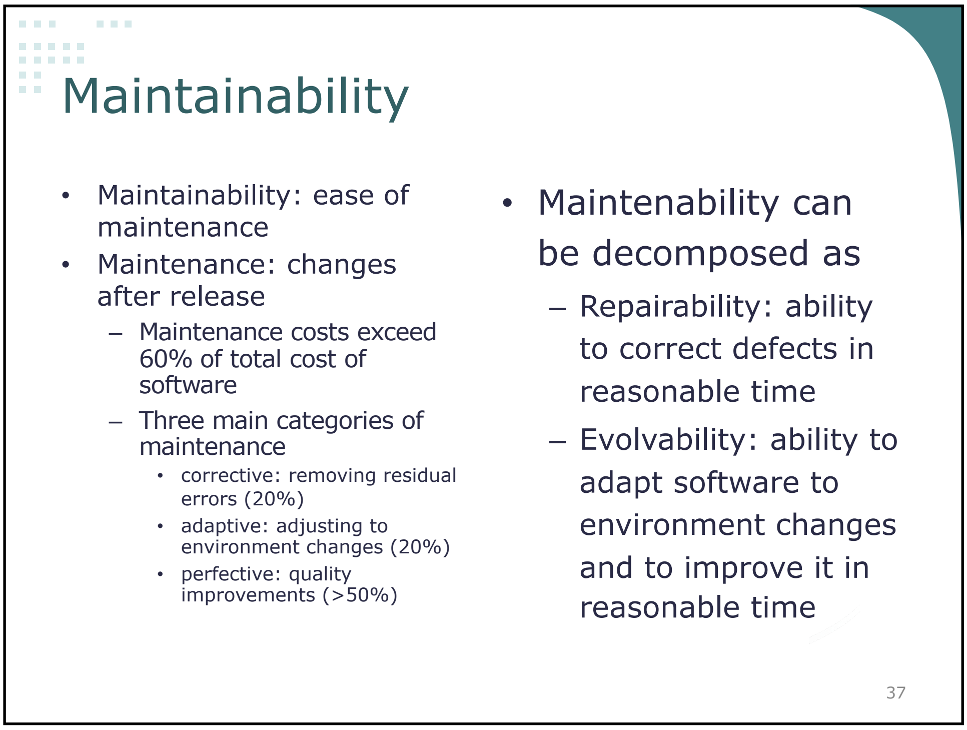
I. Software engineering
II. Software quality
1. Classification
2. Representative qualities
2.1. Common qualities

Verifiability

- How easy it is to verify properties
  - mostly an internal quality
  - can be external as well (e.g., security critical application)

36

36



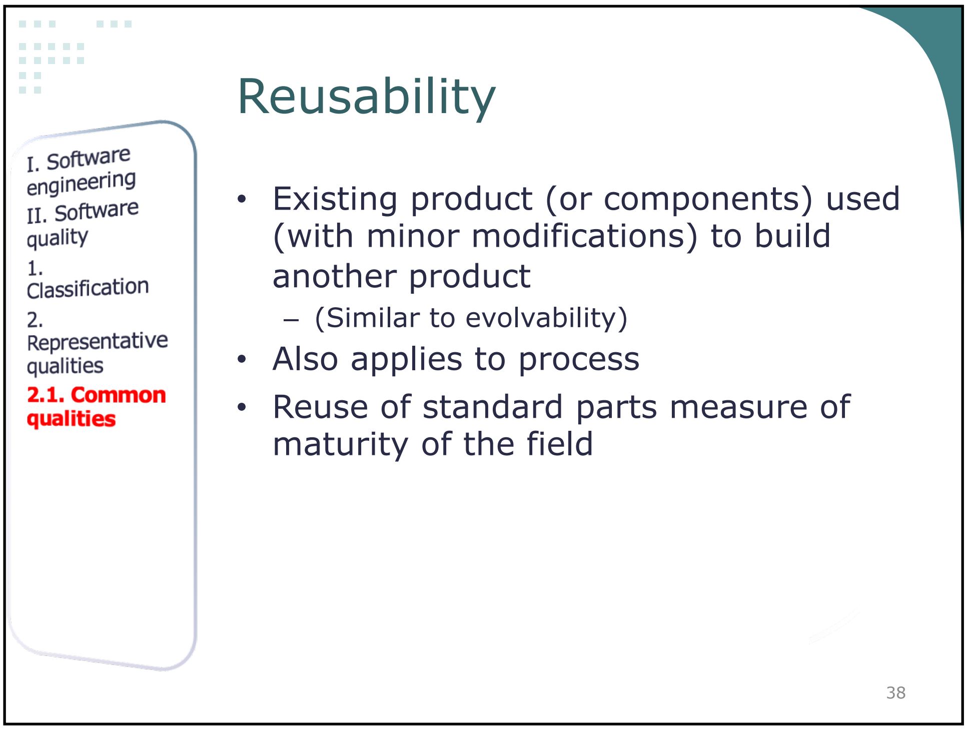
Maintainability

- Maintainability: ease of maintenance
- Maintenance: changes after release
  - Maintenance costs exceed 60% of total cost of software
  - Three main categories of maintenance
    - corrective: removing residual errors (20%)
    - adaptive: adjusting to environment changes (20%)
    - perfective: quality improvements (>50%)

- Maintenability can be decomposed as
  - Repairability: ability to correct defects in reasonable time
  - Evolvability: ability to adapt software to environment changes and to improve it in reasonable time

37

37



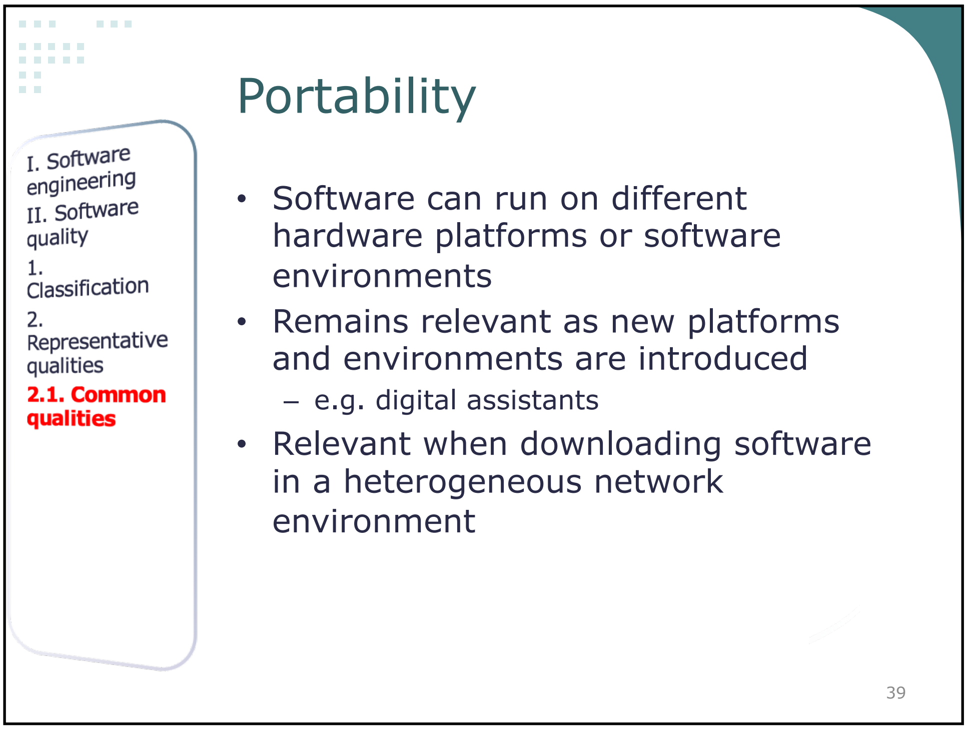
I. Software engineering
II. Software quality
1. Classification
2. Representative qualities
2.1. Common qualities

Reusability

- Existing product (or components) used (with minor modifications) to build another product
  - (Similar to evolvability)
- Also applies to process
- Reuse of standard parts measure of maturity of the field

38

38



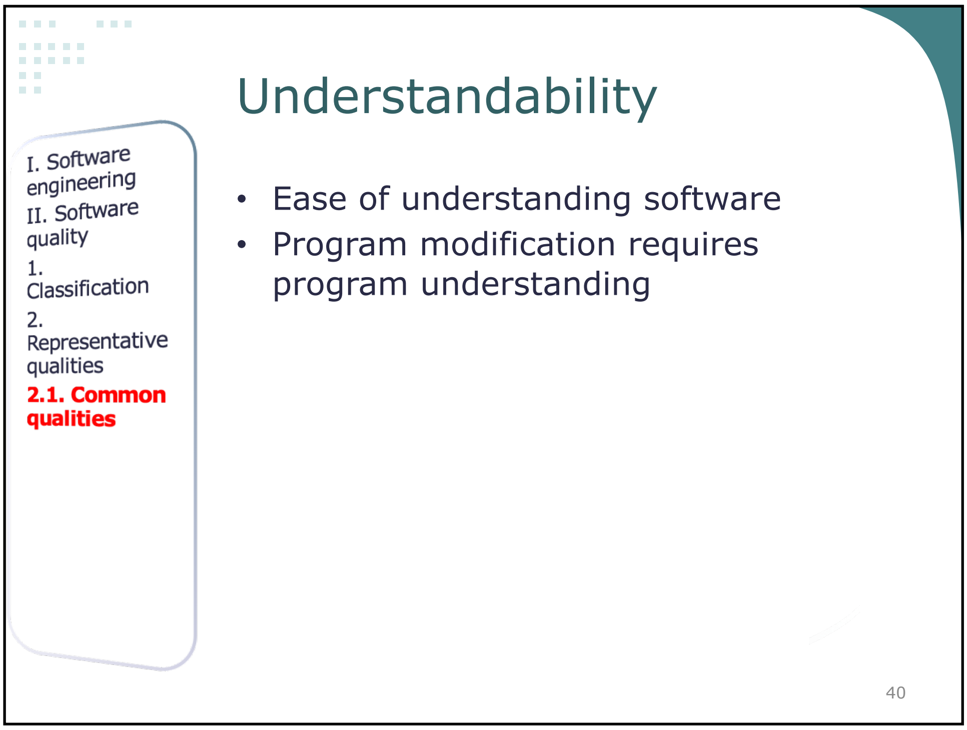
I. Software engineering
II. Software quality
1. Classification
2. Representative qualities
2.1. Common qualities

Portability

- Software can run on different hardware platforms or software environments
- Remains relevant as new platforms and environments are introduced
  - e.g. digital assistants
- Relevant when downloading software in a heterogeneous network environment

39

39

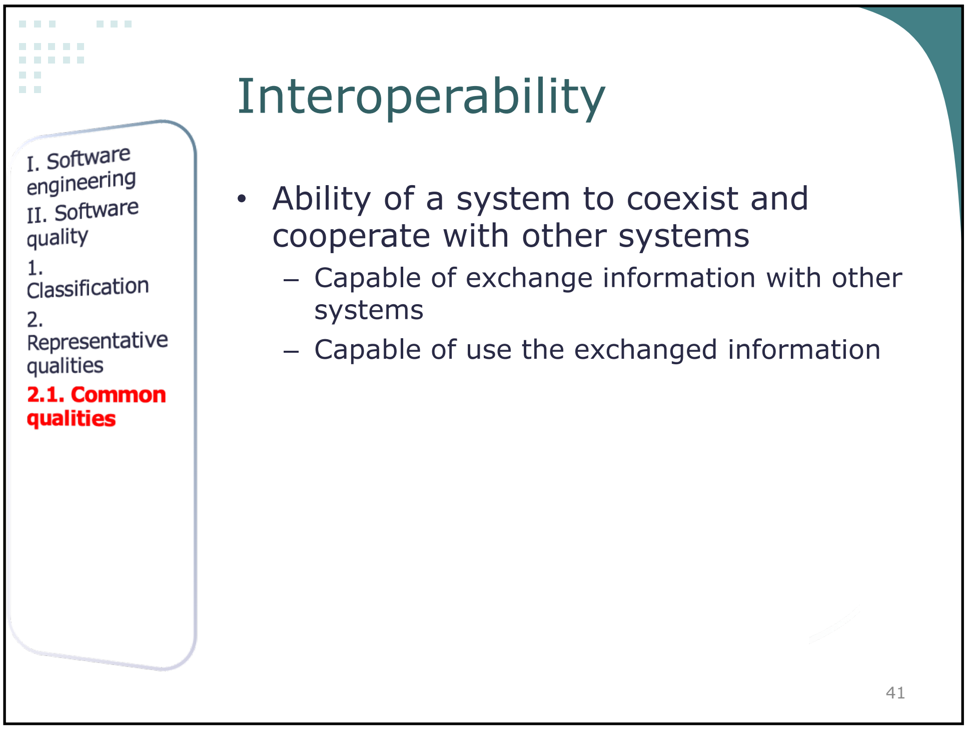


I. Software engineering
II. Software quality
1. Classification
2. Representative qualities
2.1. Common qualities

Understandability

- Ease of understanding software
- Program modification requires program understanding

40



I. Software engineering
II. Software quality
1. Classification
2. Representative qualities
2.1. Common qualities

Interoperability

- Ability of a system to coexist and cooperate with other systems
  - Capable of exchange information with other systems
  - Capable of use the exchanged information

41