AGILE SOFTWARE ENGINEERING:

QUALITY MANAGEMENT

JOHN STOUBY PERSSON

AALBORG UNIVERSITY
DENMARK

Last lecture’s objectives

Knowledge about risks in software engineering

Skills in software risk identification, assessment, mitigation, monitoring, and management.

Competencies to manage risks in software engineering.

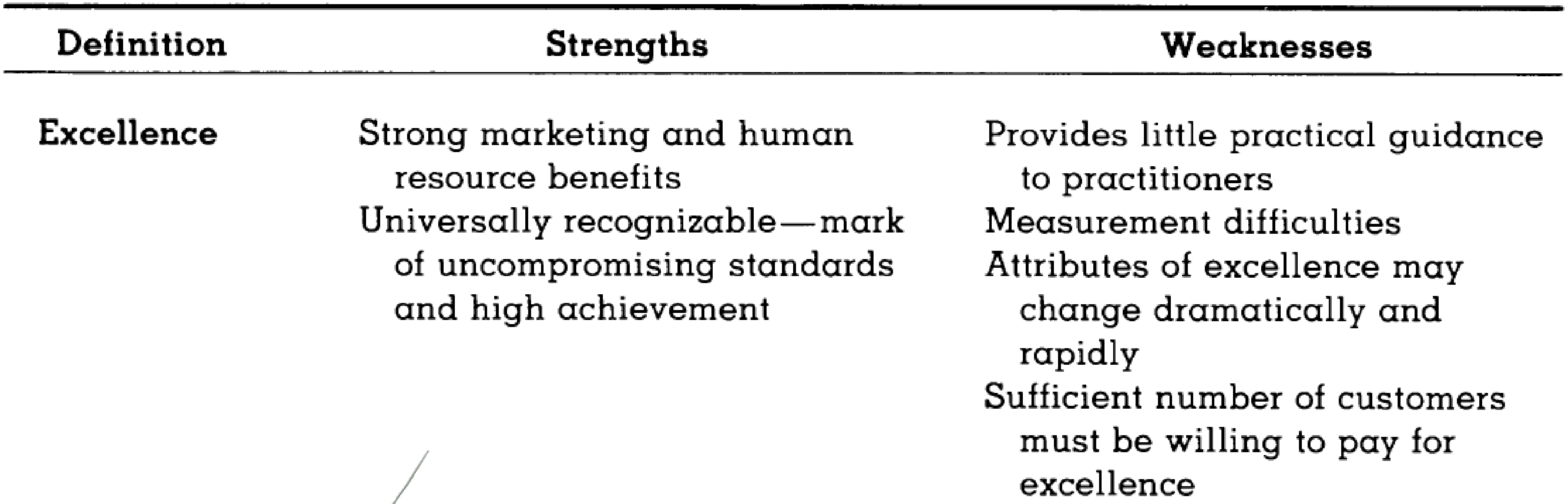
Lecture objectives

Knowledge about the quality problems in software engineering

Skills in defining and improving the quality of a software product through its related artifacts and processes.

Competencies to manage quality in agile software engineering.

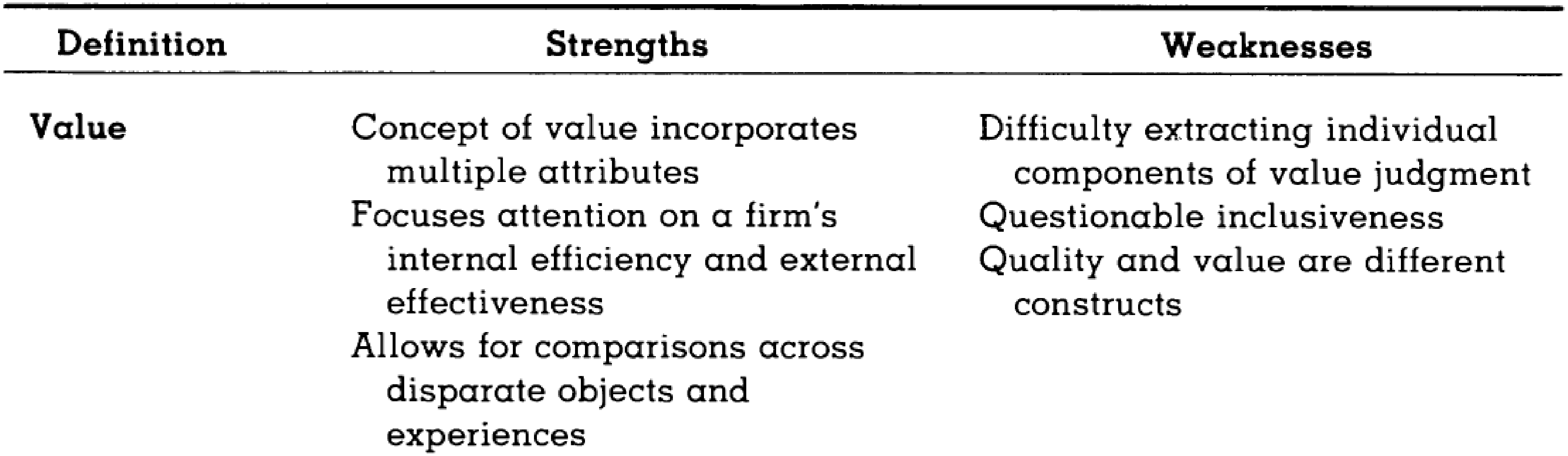
The problem of defining quality: Quality as Excellence



Definition	Strengths	Weaknesses
Excellence	Strong marketing and human resource benefits Universally recognizable—mark of uncompromising standards and high achievement	Provides little practical guidance to practitioners Measurement difficulties Attributes of excellence may change dramatically and rapidly Sufficient number of customers must be willing to pay for excellence

Reeves, C.A. & Bednar, D.A. "Defining Quality: Alternatives and Implications," *Academy of Management Review*, (19:3), 1994, pp. 419-445.

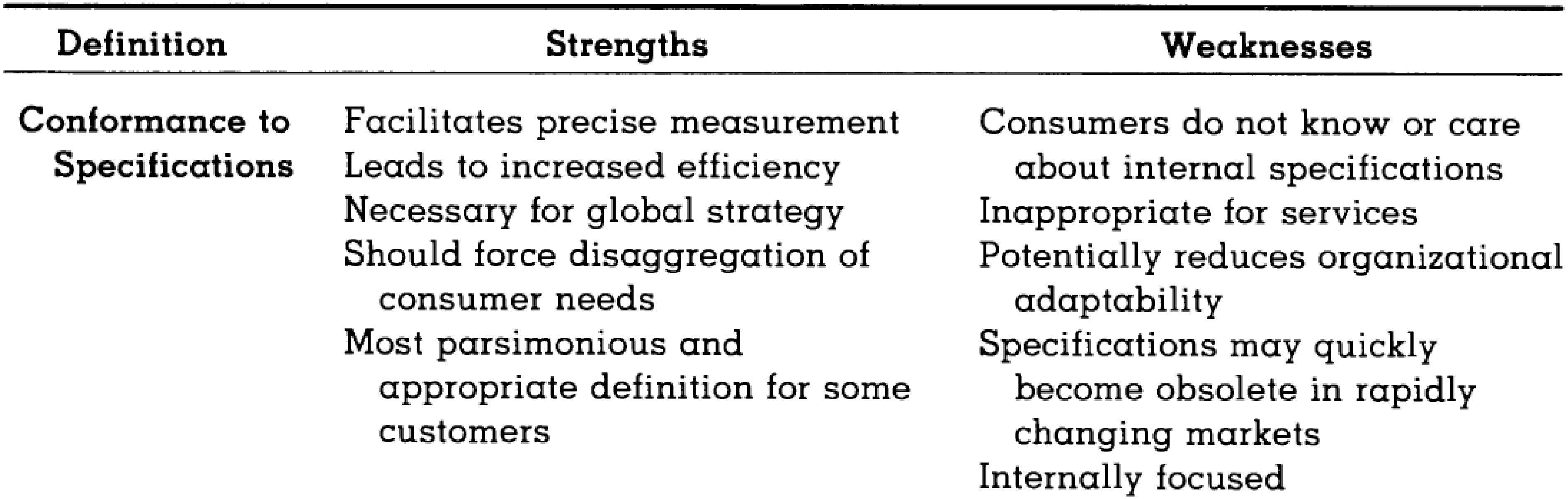
The problem of defining quality: Quality as Value



Definition	Strengths	Weaknesses
Value	Concept of value incorporates multiple attributes Focuses attention on a firm's internal efficiency and external effectiveness Allows for comparisons across disparate objects and experiences	Difficulty extracting individual components of value judgment Questionable inclusiveness Quality and value are different constructs

Reeves, C.A. & Bednar, D.A. "Defining Quality: Alternatives and Implications," *Academy of Management Review*, (19:3), 1994, pp. 419-445.

The problem of defining quality: Quality as Conformance



Definition	Strengths	Weaknesses
Conformance to Specifications	Facilitates precise measurement Leads to increased efficiency Necessary for global strategy Should force disaggregation of consumer needs Most parsimonious and appropriate definition for some customers	Consumers do not know or care about internal specifications Inappropriate for services Potentially reduces organizational adaptability Specifications may quickly become obsolete in rapidly changing markets Internally focused

Reeves, C.A. & Bednar, D.A. "Defining Quality: Alternatives and Implications," *Academy of Management Review*, (19:3), 1994, pp. 419-445.

The problem of defining quality: Quality as Expectations

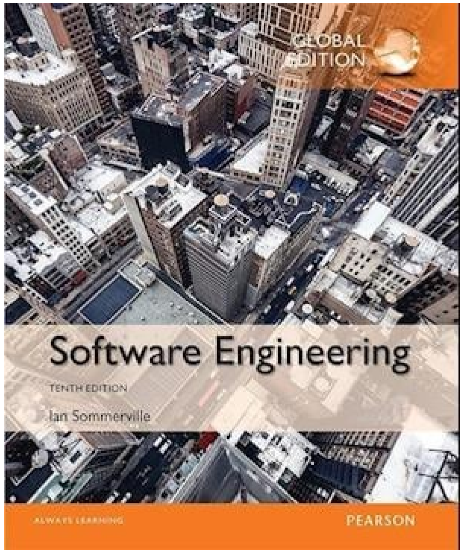
Definition	Strengths	Weaknesses
Meeting and/or Exceeding Expectations	Evaluates from customer's perspective Applicable across industries Responsive to market changes All-encompassing definition	Most complex definition Difficult to measure Customers may not know expectations Idiosyncratic reactions Pre-purchase attitudes affect subsequent judgments Short-term and long-term evaluations may differ Confusion between customer service and customer satisfaction

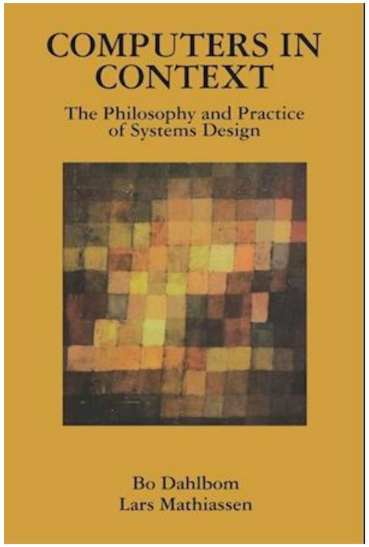
Reeves, C.A. & Bednar, D.A. "Defining Quality: Alternatives and Implications," *Academy of Management Review*, (19:3), 1994, pp. 419-445.

Short discussion

What definition of quality dominates in your project?

Quality as 1.Excellence, 2. Value, 3. Conformance to specifications, OR 4. Meeting and/or exceeding expectations.





Definition of software quality

Quality is a reflection of one or more peoples’ assessment of correspondance between their expectations and experience of a product or service

Quality can be divided into three types:

- Product
- Process
- Expectations

These 3 types of quality drive a mix of expectations to

- Functionality often described with functional requirements/userstories
- Non-functional requirements often called Software Quality Attributes or Software Quality Factors (e.g. efficient, usable, …)
- How the product is created or product quality is ensured (process)

Expectations to quality from requirements

Functional requirements are often described in

- Requirement Specification (e.g. waterfall)
- Product backlog and User Stories (e.g. agile)

Non-functional Requirements are described as different categories of software quality attributes:

Safety	Understandability	Portability
Security	Testability	Usability
Reliability	Adaptability	Reusability
Resilience	Modularity	Efficiency
Robustness	Complexity	Learnability

(Sommerville)

You can’t have all Quality Attributes – tradeoffs are inevitable

🟢 **Positive influence**
🔴 **Negative influence**

	Usability	Integrity	Efficiency	Reliability	Robustness	Maintainability	Testability	Flexibility	Reuseability	Transferability	Interoperability
Usability		🟢	🔴	🟢	🟢	🟢	🟢	🟢			
Integrity	🟢		🔴					🔴	🔴		🔴
Efficiency	🔴	🔴		🔴	🔴	🔴	🔴	🔴	🔴	🔴	🔴
Correctness	🟢		🔴		🟢	🟢	🟢	🟢			
Reliability	🟢		🔴	🟢		🟢	🟢	🟢	🔴		
Maintainability	🟢		🔴	🟢	🟢		🟢	🟢	🟢	🟢	
Testability	🟢		🔴	🟢	🟢	🟢		🟢	🟢	🟢	
Flexibility	🟢	🔴	🔴	🟢	🟢	🟢	🟢		🟢	🟢	
Reuseability		🔴	🔴		🔴	🟢	🟢	🟢		🟢	
Transferability			🔴			🟢	🟢	🟢	🟢		🟢
Interoperability		🔴	🔴							🟢	

Expectations to process - Quality Management

- Quality Management consists of
  - Quality Assurance (Plan or design processes to prevent bad quality)
  - Quality Control (Monitor that work products meet quality standards)

” Software Quality Management techniques have their roots in methods and techniques that were developed in manufacturing industries, where the terms *quality assurance* and *quality control* are widely used.

Quality assurance is the definition of processes and standards that should lead to high quality products and the introduction of quality processes in the manufacturing process.

Quality control is the application of these quality processes to weed out products that are not of the required level of quality. Both quality assurance and quality control are part of quality management. ”

- Sommerville

Cost of quality management

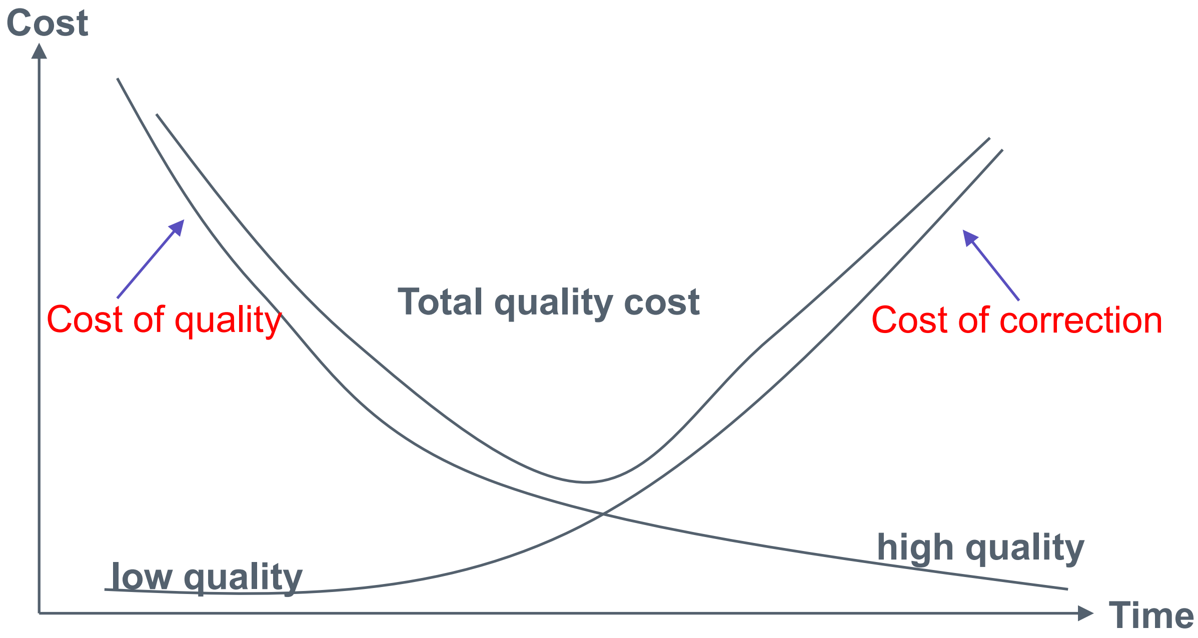
Preventive costs

- Better planning (of review, test and so on)
- Measurable quality factors (e.g. response time, load, ..)
- Education and training of developers (in test, review, ..)

Monitoring costs

- Evaluation, peer review, code inspection (do planned reviews)
- Testing (do planned tests)

Overall theory of cost and quality



- Low quality (low quality management) is initially cheap, but becomes gradually more expensive
- High quality management, has an initial cost when quality processes are defined, but is cheaper later because users are reporting much less errors, and the code is more stable
- Quality management should be balanced to the cost – a process that is 100% defect free is very expensive, while decreasing quality management, will increase the amount of defects reported over time.
- We plan and design, how and when to do *verification* and *validation* in our process

Quality Assurance: quality of product and expectations

NEJ JA

Validation: (fit for use)

- Are we building a system that is fit for use?
- Conforms to customers’ expectations and experience

Verification: (Are all requirements implemented)

- Are we building the system with all requirements implemented
- Conforms to requirement specification

Techniques for verification and validation

In general the techniques are

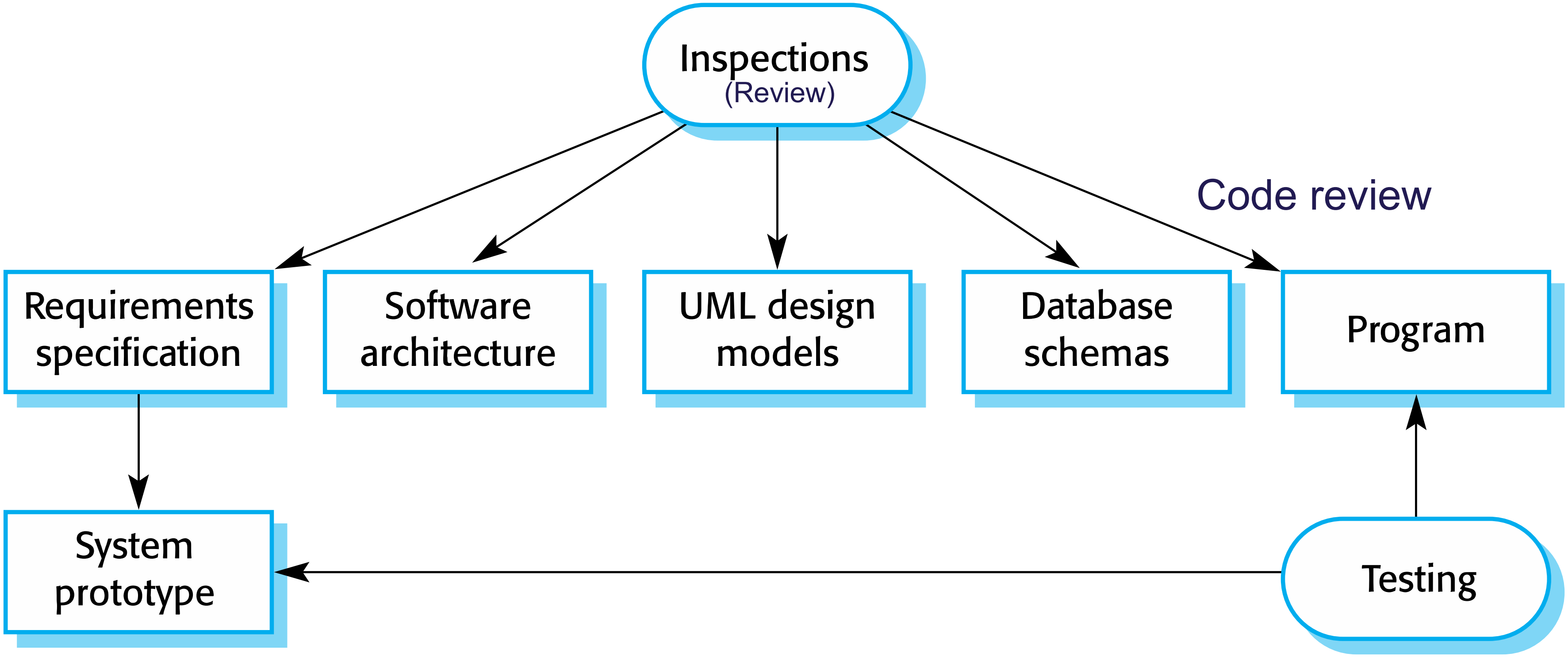
- Testing of programs and prototypes
- Reviewing of specifications, documentation and programs

Is it verification or validation?

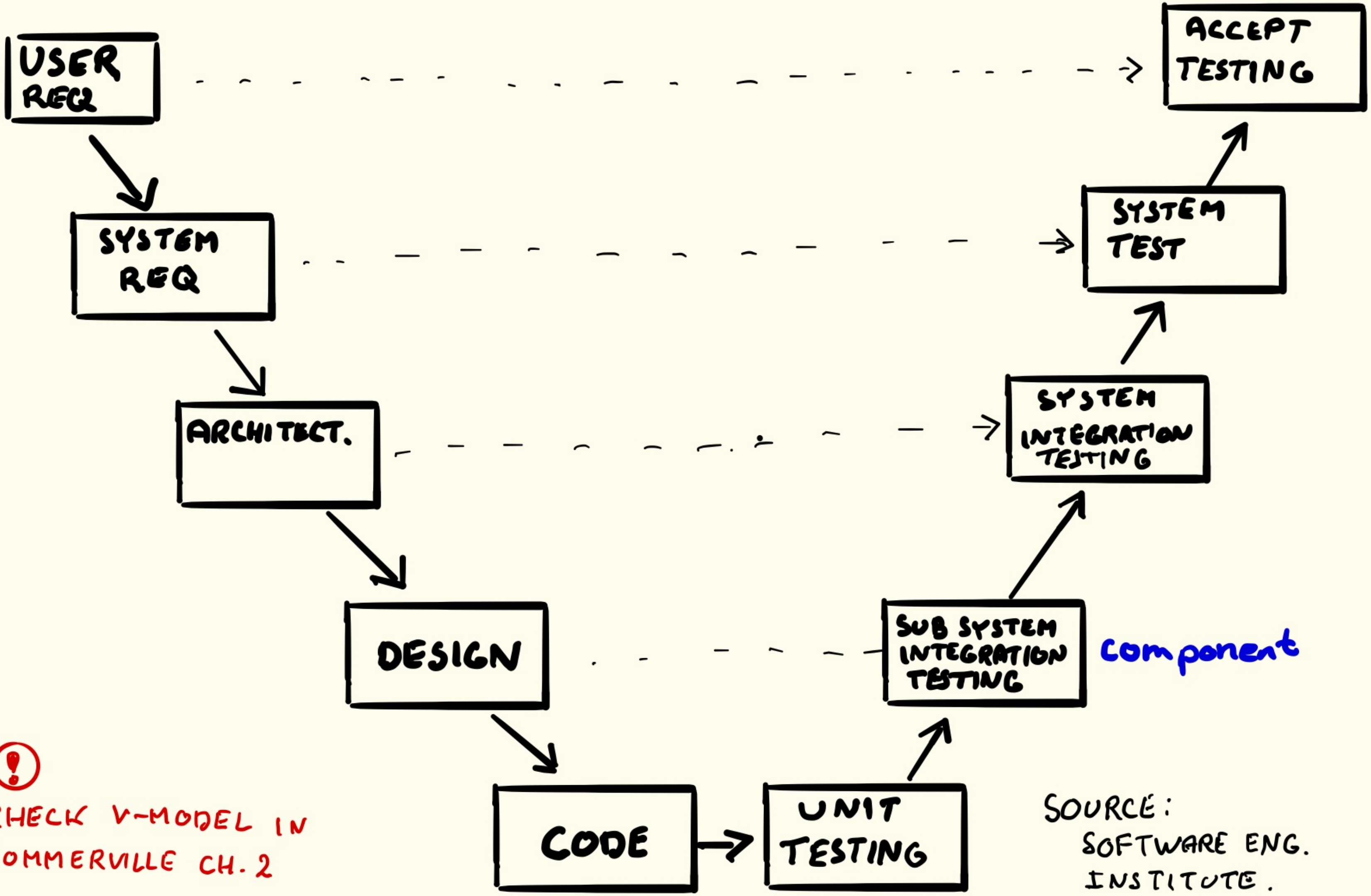
- for a test or review to be validation a user must participate, since they know if something is “fit for use”.
- a verification activity will focus on compliance to specification, and typically a user and customer do not participate.

Inspections and testing

- Inspections (Reviews) and testing are complementary and should be used during the Verification & Validation process.
- Inspections (reviews) can check conformance with a specification but not conformance with the customer's or users real requirements (non-functional characteristics). Users can be involved with in a prototype instead of a review.



V-MODEL

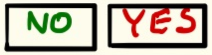


! CHECK V-MODEL IN SOMMERVILLE CH. 2

SOURCE:
SOFTWARE ENG.
INSTITUTE.

TEST FOCUS

1 ACCEPT

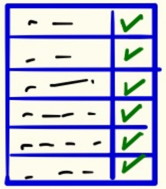


FIT FOR USE

EXPLORATORY

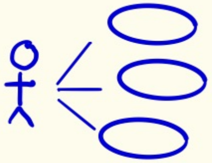
EXPECTATION

2 SYSTEM



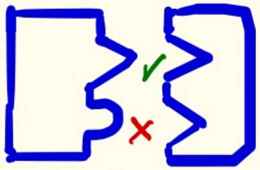
ALL REQIREMENTS

PERFORMANCE



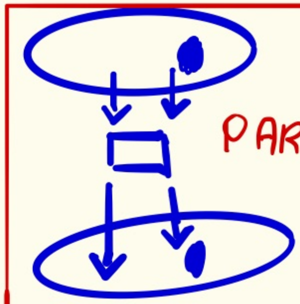
USE CASE BASED

3 SUB SYSTEM INTEGRATION



INTERFACES

4 UNIT

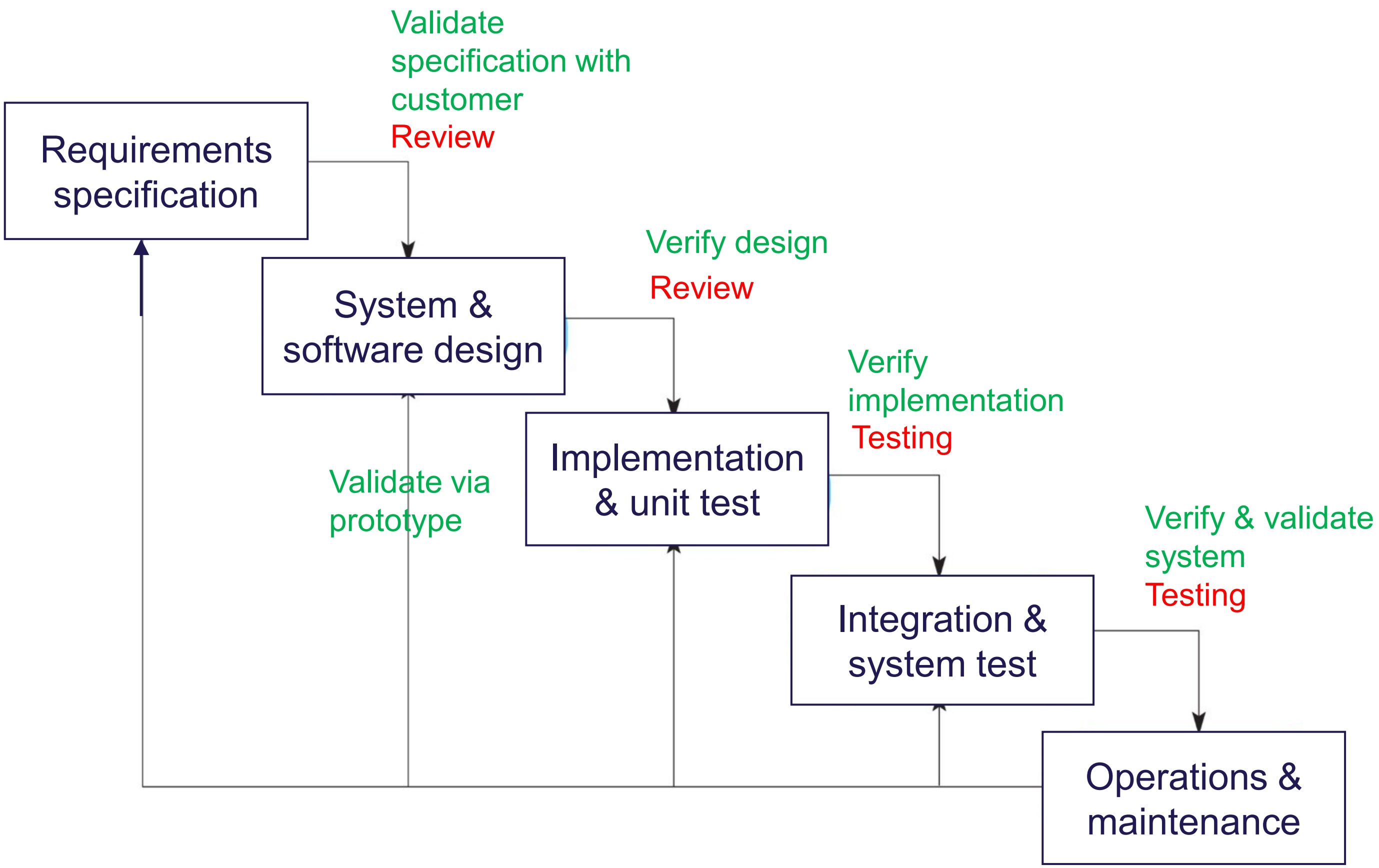


PARTITION

<4	4-10	>10

INVALID INPUT

Waterfall Model



Requirements specification

Validate specification with customer
Review

System & software design

Verify design
Review

Validate via prototype

Implementation & unit test

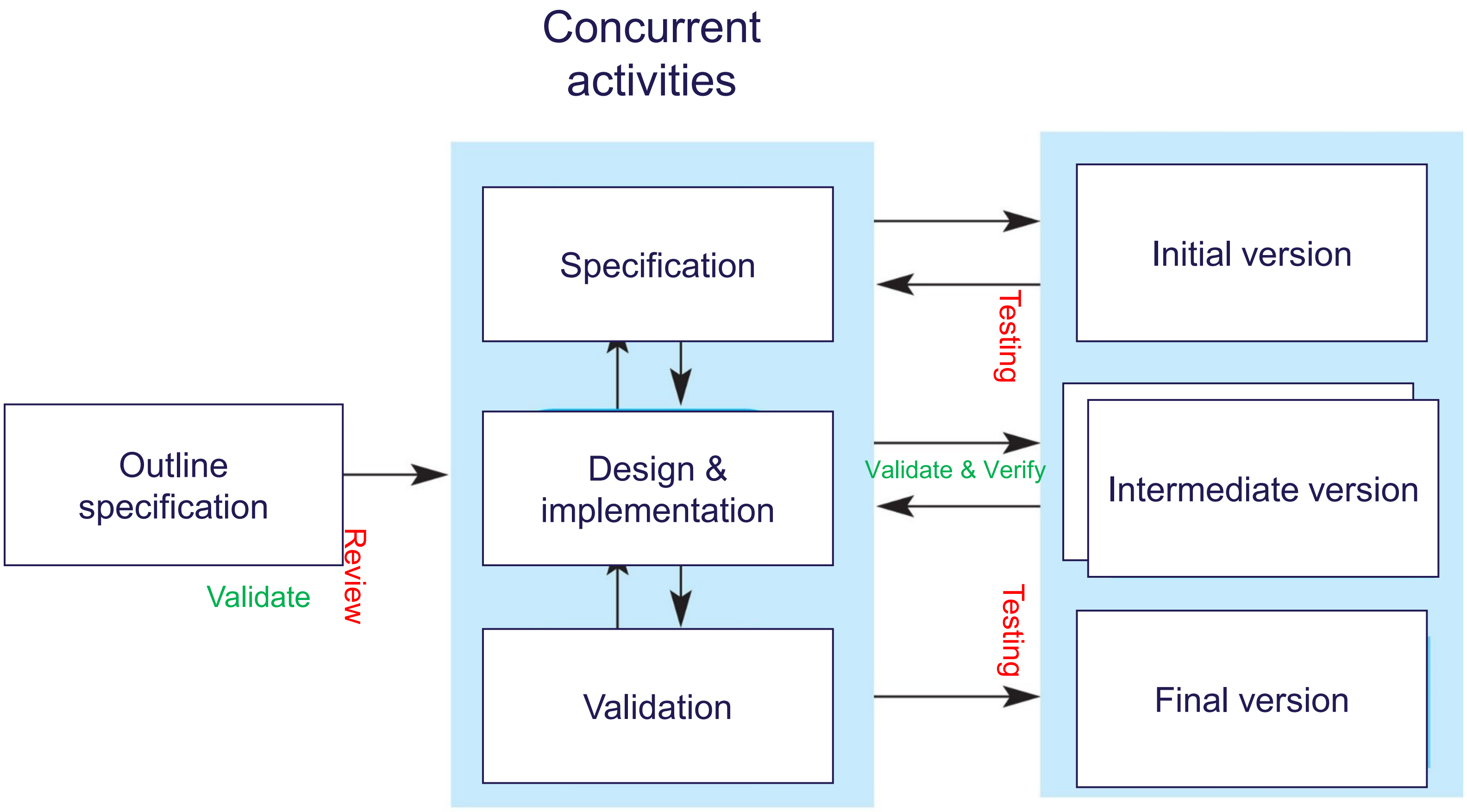
Verify implementation
Testing

Integration & system test

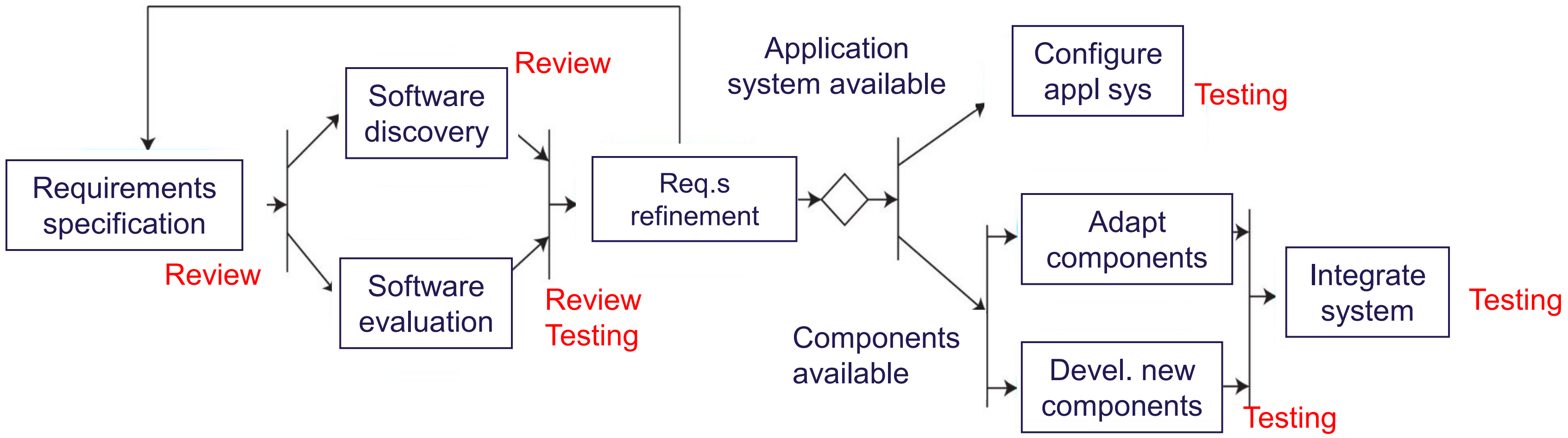
Verify & validate system
Testing

Operations & maintenance

Incremental Model



Integration & configuration



Requirements specification

Review

Software discovery

Review

Software evaluation

Review

Testing

Req.s refinement

Application system available

Configure appl sys

Testing

Components available

Adapt components

Devel. new components

Testing

Integrate system

Testing

Quality management and agile development

Informal rather than document-based by stablishing a **quality culture**, where all team members are responsible for software quality through **agile quality practices**:

Definition of Done: Team agree on criteria for a task to be complete

Sprint review: PO and other stakeholders validate the sprint delivery meets expectations

Check before check-in: Developers are responsible for organizing their own code reviews with other team members before the code is checked in to the build system.

Never break the build: Team members should not cause the system to fail by testing their code changes against the whole system before check-in.

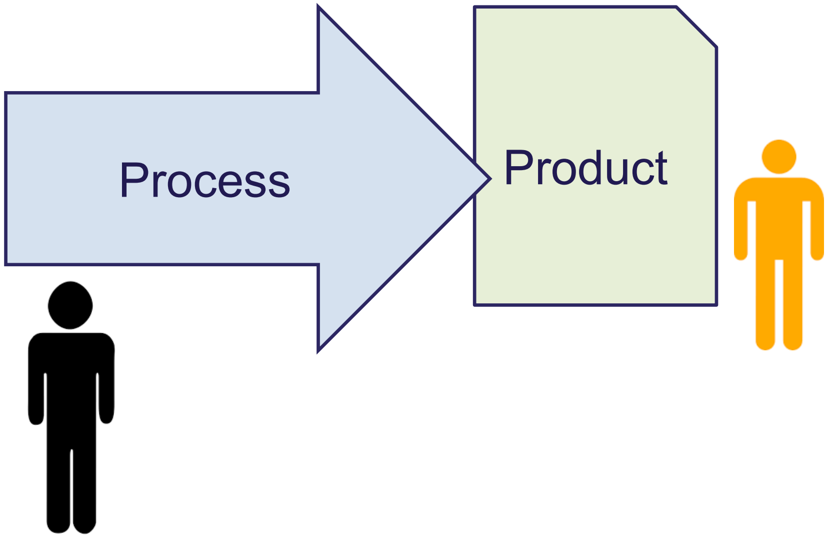
Fix problems when you see them: If a programmer discovers problems or obscurities in code developed by someone else, they can fix these directly rather than referring them back to the original developer.

Fundamental Process Theory (Process Quality)

A software product can only be as good as the process through which it is produced.

You can only improve the quality of the product if you improve the process

- Repeating the same process, will create same level of quality
- Sources of bad quality can be used as input to improve the process



Group exercises

Exercise 1: Understanding quality in your project (50%)

Exercise 2: Quality management in your project (50%)

Lecture objectives

Knowledge about the quality problems in software engineering

Skills in defining and improving the quality of a software product through its related artifacts and processes.

Competencies to manage quality in agile software engineering.