

Requirements Engineering I

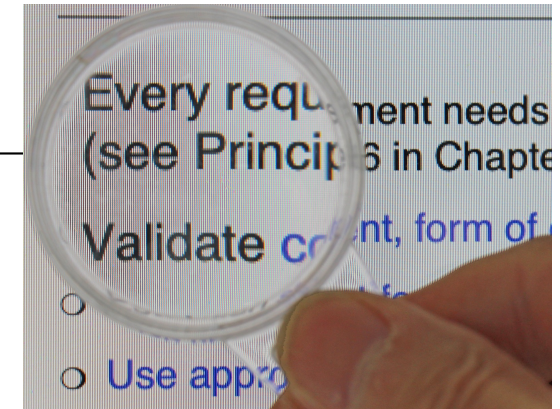
Chapter 9

Validating Requirements



9.1 What to validate

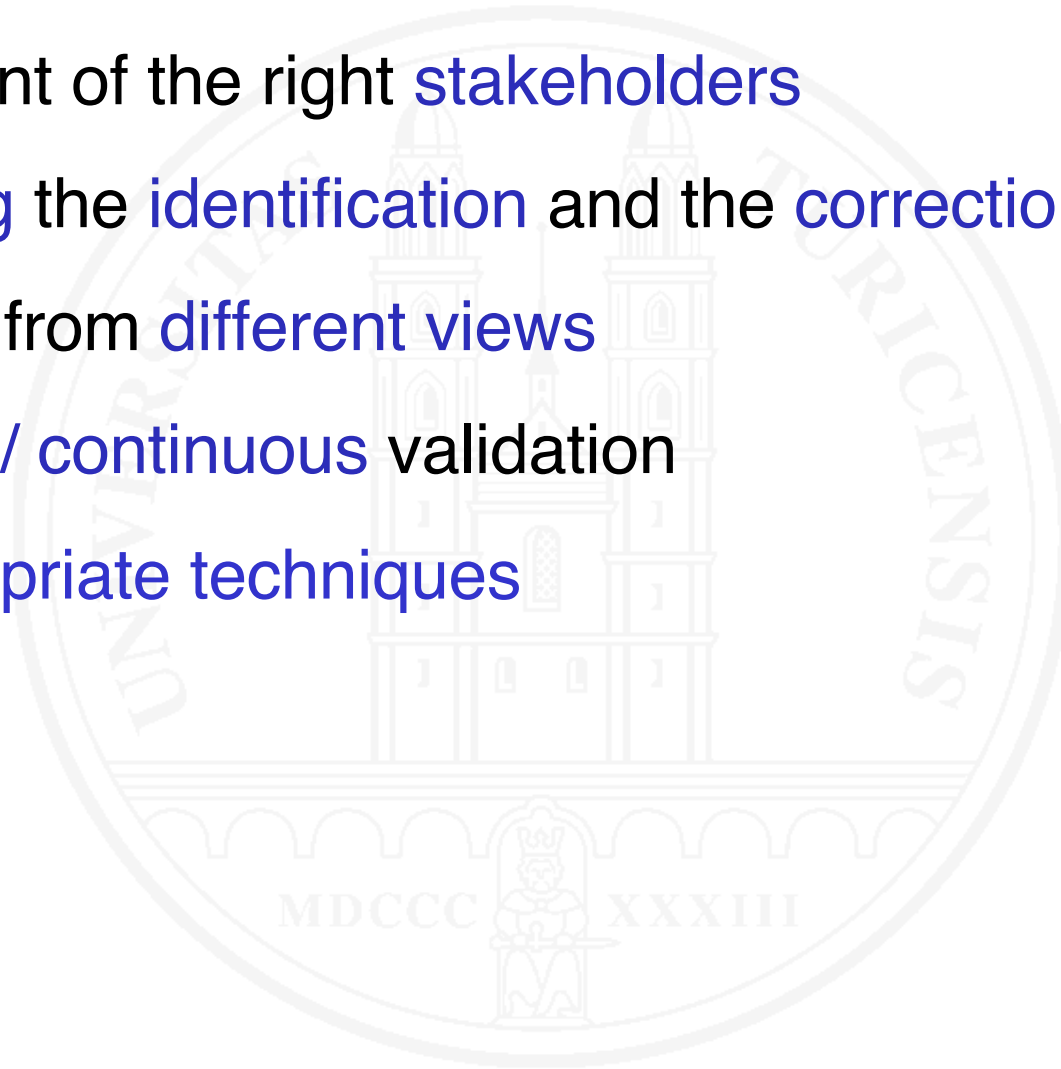
Every requirement needs to be validated
(see Principle 6 in Chapter 2)



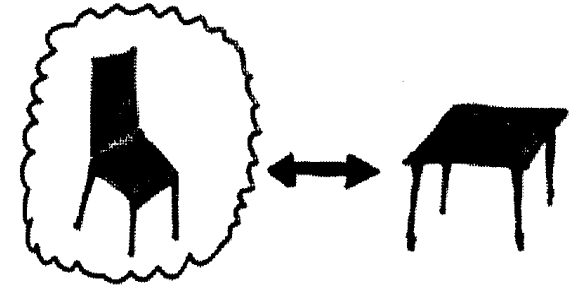
- Content:
 - Stakeholders' desires and needs adequately covered?
 - Requirements agreed?
- Work products: Requirements documented well?
- Context: Assumptions reasonable?

Important validation aspects

- Involvement of the right stakeholders
- Separating the identification and the correction of defects
- Validation from different views
- Repeated / continuous validation
- Use appropriate techniques



Validation of content



Identify requirements that are

- Inadequate or wrong
- Incomprehensible
- Incomplete or missing
- Ambiguous

Also look for requirements with these quality defects:

- Not verifiable
- Unnecessary
- Infeasible
- Premature design decisions



Validation of content: Agreement

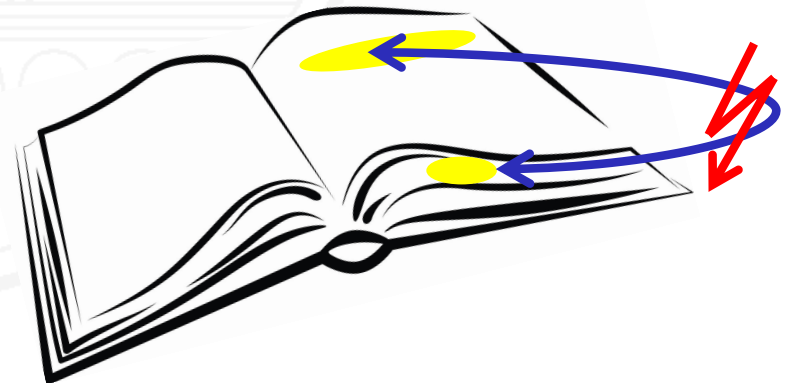
- Requirements elicitation involves achieving **consensus** among stakeholders having divergent needs
- When validating requirements, we have to check whether **agreement** has actually been **achieved**
 - All known **conflicts negotiated** and **resolved**? → Chapter 4.4
 - For all requirements: have all relevant stakeholders for a requirement **agreed** to this requirement in its **documented form**?
 - For every **changed** requirement, have all relevant stakeholders **agreed to this change**?

Validation of requirements work products

Scope: checking the requirements **work products** (e.g., a systems requirements specification or a collection of user stories) for formal problems

Identify requirements that are

- **Inconsistent** with each other
- **Missing**
- **Non-conforming** to documentation **rules**, **structure** or **format**
- **Redundant**
- **Badly structured**
- **Hard to modify**
- **Not traceable**



Context validation

- Context assumptions reasonable?
- Mappings from context phenomena to system inputs / outputs adequate?
- Can we reasonably argue that the domain requirements will be met when the system will be built and deployed as specified in the requirements?

9.2 Requirements validation techniques

Review

- Main means for requirements validation
- **Walkthrough**: author guides experts through the specification
- **Inspection**: Experts check the specification
- **Author-reviewer-cycle**: Requirements engineer continuously feeds back requirements to stakeholder(s) for review and receives feedback

Construction of other work products

- **Acceptance criteria / test cases** help disambiguate / clarify requirements
- Writing **user manuals** or creating **models for textual requirements** may help identify missing or wrong requirements

Requirements validation techniques – 2

Prototyping

- Lets stakeholders judge the practical usefulness of the specified system in its real application context
- Prototype constitutes a sample model for the system-to-be
- Most powerful, but also most expensive means of requirements validation

Simulation/Animation

- Means for investigating dynamic system behavior
- Simulator executes specification and may visualize it by animated models

Requirements validation techniques – 3

Testing (when evolving an existing system)

- A/B testing
- Classic alpha and beta testing of source code

Requirements Engineering tools

- Help find gaps and contradictions

Formal Verification / Model Checking / Model Analysis

- Formal proof of critical properties
- Automated, systematic and comprehensive test of critical properties (when proofs are not tractable)

Reviewing practices

- Paraphrasing
 - Explaining the requirements in the reviewer's own words
- Perspective-based reading
 - Analyzing requirements from different perspectives, e.g., end-user, tester, architect, maintainer,...
- Playing and executing
 - Playing scenarios
 - Mentally executing acceptance test cases
- Checklists
 - Using checklists for guiding and structuring the review process

Mini-Exercise

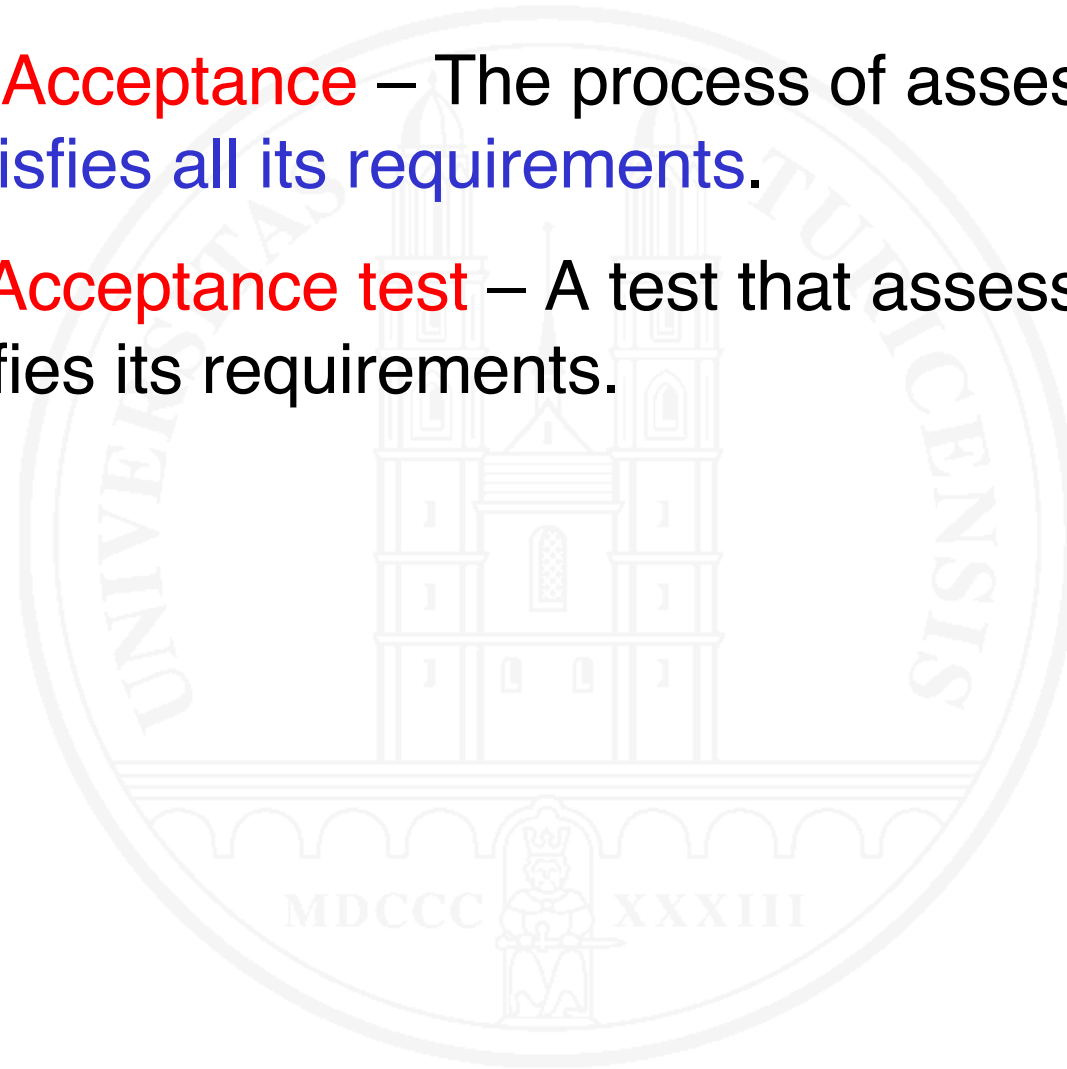
Consider the chairlift access control case study. How can you validate the following requirements (1) during RE, (2) prior to the deployment of the system?

- (a) “The system shall prevent skiers from using the chairlifts without a valid ticket.”
- (b) “The reaction time from sensing a valid card to issuing an ‘unlock for a single turn’ command shall be shorter than 0.5 s.”
- (c) “As a skier, I want to load a ticket that I bought online to my access card, so that it is fast and convenient for me and I do not have to queue at a ticket counter.”

9.3 The role of acceptance testing

DEFINITION. **Acceptance** – The process of assessing whether a system **satisfies all its requirements**.

DEFINITION. **Acceptance test** – A test that assesses whether a system satisfies its requirements.



Requirements and acceptance testing

Requirements engineering and acceptance testing are naturally **intertwined**

- For every requirement, there should be **at least one acceptance test case**
- Requirements should be written such that acceptance tests can be written to **verify** them (→ verifiability)
- Acceptance test cases can serve
 - for **disambiguating** requirements
 - as **detailed specifications** by example → acceptance criteria for user stories

Choosing acceptance test cases

Potential coverage criteria:

- **Requirements** coverage: At least one case per requirement
- **Function** coverage: At least one case per function
- **Scenario** coverage: For every type scenario / use case
 - All actions covered
 - All branches covered
- Consider the **usage profile**: not all functions/scenarios are equally frequent / important