Software Requirements and Architecture (SENG404)

Matthias Galster

Lecture 7 – Requirements validation

March 22, 2023

UC
UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Schedule 2023

Lecture	Week	Date	Topic
1	1	February 22	Kick-off; Introduction
2	1	February 23	Instead of May 3; Requirements and requirements engineering processes
3	2	March 1	Requirements elicitation (part 1)
4	2	March 2	Instead of May 17; Requirements elicitation (part 1); Requirements elicitation (part 2)
5	3	March 8	Requirements elicitation (part 2); Requirements documentation
6	3	March 9	Matthias away
7	4	March 15	Requirements documentation
8	4	March 16	Requirements documentation; Requirements analysis
9	5	March 22	Assignment 1; Requirements analysis, Requirements validation
10	6	March 29	
Term break			
11	7	April 26	
12	8	May 3	Matthias away
13	9	May 10	
14	10	May 17	Matthias away
15	11	May 24	Assignment 2: presentations + report
16	12	May 31	
		TBD	**Final exam**

Assignment 1

Student(s)	Topic
Saskia van der Peet	Use of design thinking in requirements engineering
April Clarke	Influence of social factors on requirements engineering
Jonathan Tomlinson + Danish Jahangir	Software requirements elicitation techniques
Michael Wilson	Architecture recovery and recovery techniques
Jamie Thomas	Usefulness in requirements prioritization techniques
Lisa Lu + Joshua Egan	Towards Understanding Software Architectural Patterns
Andrew Cook	Security requirements engineering

Previous lecture

1. Requirements analysis – overview

2. Effort estimation

3. Requirements prioritization

Reading for this session

- L. Montgomery, D. Fucci, A. Bouaffa, L. Scholz, and W. Maalej. *Empirical research on requirements quality: a systematic mapping study.* Requirements Engineering, 2022, pp. 183-209, https://doi.org/10.1007/s00766-021-00367-z

Questions and lessons

- https://jamboard.google.com/d/1Hn9QmwqDMsqWBr_4FI47UzOtA02smqwH6-Dy1A6Ju2g/edit?usp=sharing

Reading for next session

- S. Jayatilleke and R. Lai. *A systematic review of requirements change management.* Information and Software Technology, 2018, pp. 163-185, doi: 10.1016/j.infsof.2017.09.004

Agenda

1. Requirements validation – overview

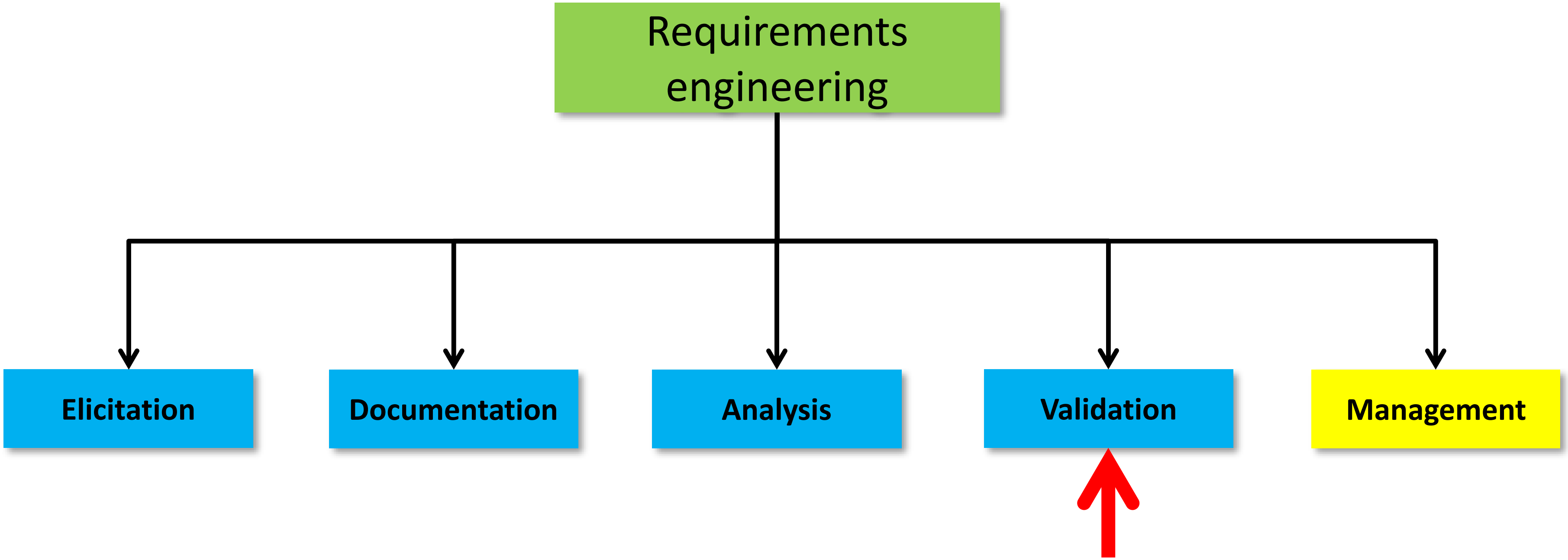
2. Requirements validation techniques

Agenda

1. Requirements validation – overview

2. Requirements validation techniques

Generic RE process activities



Requirements validation

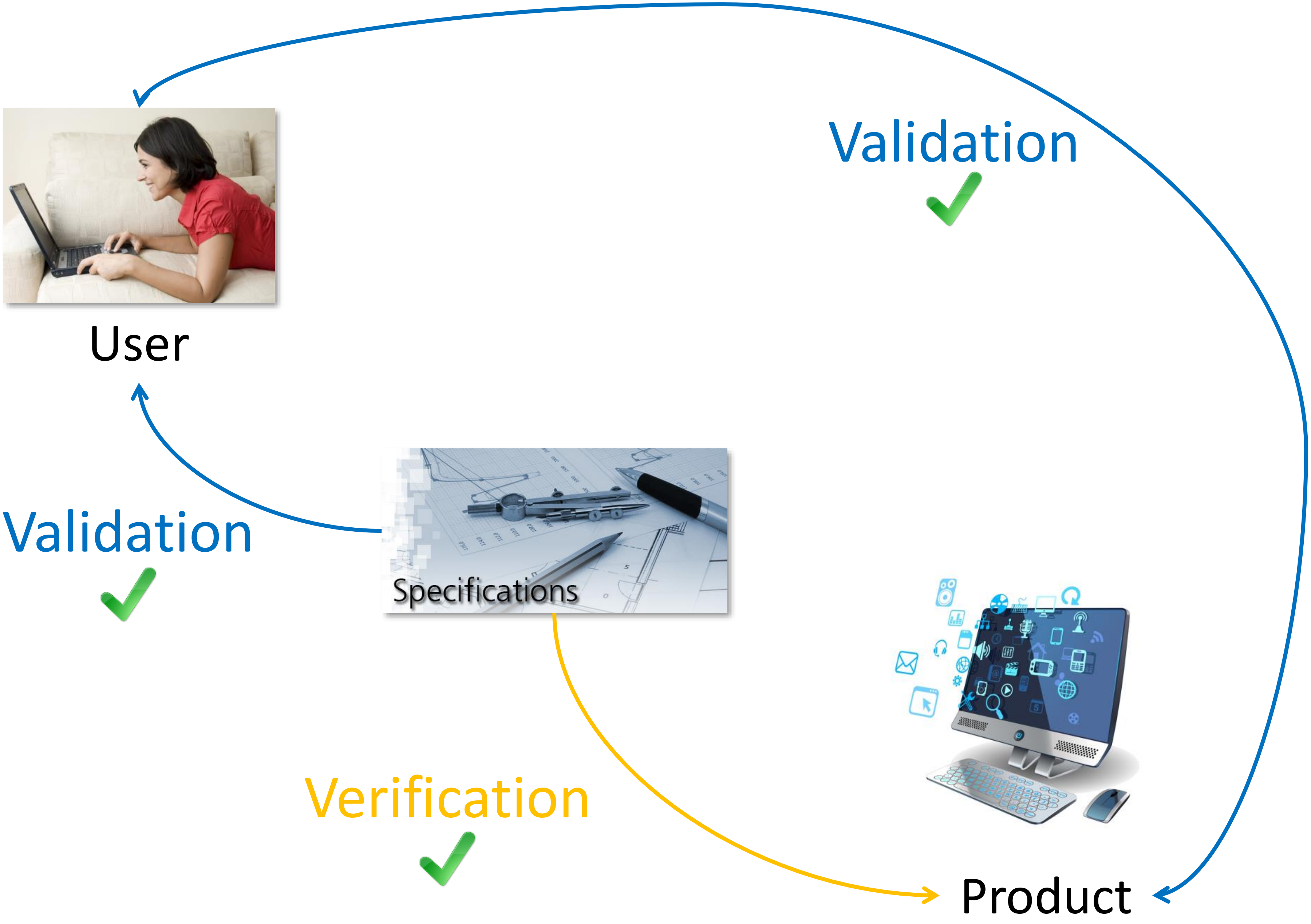
- Process and activities related to
  - ensuring that “requirements” are
  - [complete,] clear, correct, consistent, realistic

- Validation versus verification

list of requirmentents

validating, are we actually doing

verification, whether the software builds and complys



User

Validation ✓

Validation ✓

Specifications

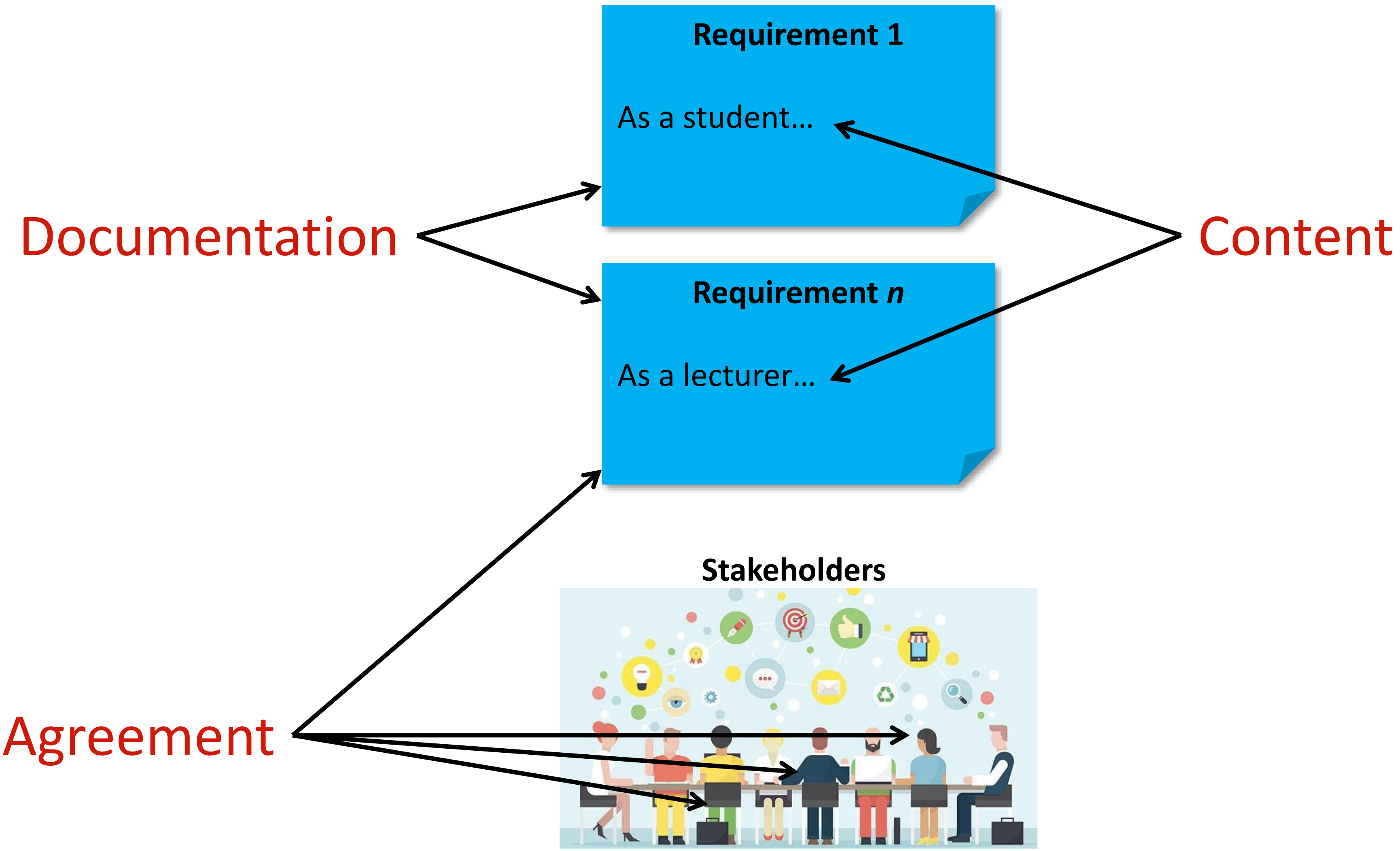
Verification ✓

Product

Validation: Are we building the right product? “User-oriented”

Verification: Are we building the product right? “Specification-oriented” (specification as in requirements documentation/description)

What to validate



Documentation

Requirement 1

As a student…

Requirement *n*

As a lecturer…

Content

Stakeholders

Agreement

What to validate

- Content
- Documentation
- Agreement

Validation of content

- Identify requirements that have “quality defects”
  - See previous lectures (e.g., on requirements documentation)

- Noise
  - No information relevant to any feature
- Silence
  - Feature that is not covered by any text
- Over-specification
  - Feature of solution, rather than the problem
- Contradiction
  - Feature defined in various incompatible ways
- Ambiguity
  - Interpreted in at least two different ways
- Forward reference
  - Text that refers to a feature yet to be defined
- Wishful thinking
  - Feature that cannot possibly be validated

- Jigsaw puzzles
  - Requirements distributed/cross-referenced across document
- Duckspeak requirements
  - Only there to conform to standards
- Unnecessary invention of terminology
  - E.g., ‘the user input presentation function’
- Inconsistent terminology
  - Inventing and then changing terminology
- Problems for the development staff
  - Making reader work hard to decipher intent
- Writing for the hostile reader
  - There are fewer of these than friendly readers

What to validate

- Content
- Documentation

the way they are documented, is it thorough, does it capture details

- Agreement

Validation of documentation

- Non-conforming to documentation rules, structure, format
  - Not traceable
  - Incomplete acceptance tests
  - Missing DoD
  - Missing meta-information
  - Etc.

What to validate

- Content
- Documentation
- Agreement

Validation of agreement

- Consensus among different stakeholders?
  - Check whether agreement has actually been achieved
  - Are known conflicts resolved

Conflicts

- Multiple stakeholders have different goals and requirements

- Strong conflicts (e.g., in meeting scheduler)
  - Meeting participant

    "R1: The constraint of a participant may not be disclosed to anyone else."

    ↕

  - Meeting initiator

    "R2: The meeting initiator should know the participants' constraints."

- Weak conflicts (e.g., in library system)
  - Library staff

    "R1: A borrower should return a borrowed book copy within two weeks."

    ↕

  - Borrower

    "R2: A borrower should keep a borrowed book copy as long as she needs it."

Conflict analysis: reasons for conflicts

different view points, different goals, and different understanding of the domain, d

- Subject matter conflict
  - Divergent factual needs

dfferent understanding of the domain, may misinterpret facts. eg. response time of 1s, or finacnce want response time of 1ms, but have disvissions to solve

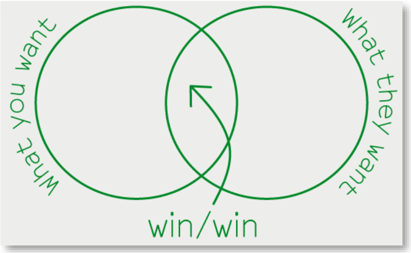
- Conflict of interest or values
  - Divergent interests, values and preferences, e.g., cost versus function

hidden agenda clients

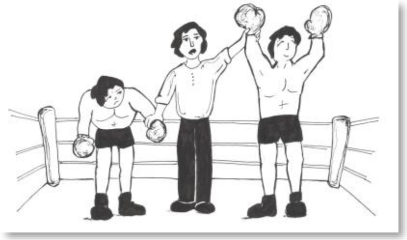
- Organizational conflict
  - Between stakeholders on different hierarchy and decision power levels
- Relationship conflict
  - Emotional problems in personal relationship between stakeholders

Conflict resolution

- Win-win techniques
  - Agreement by compromise / variants
  - Identify (win) goals, capture + reconcile differences



- Win-lose techniques
  - Overruling
  - Voting
  - Prioritizing stakeholders



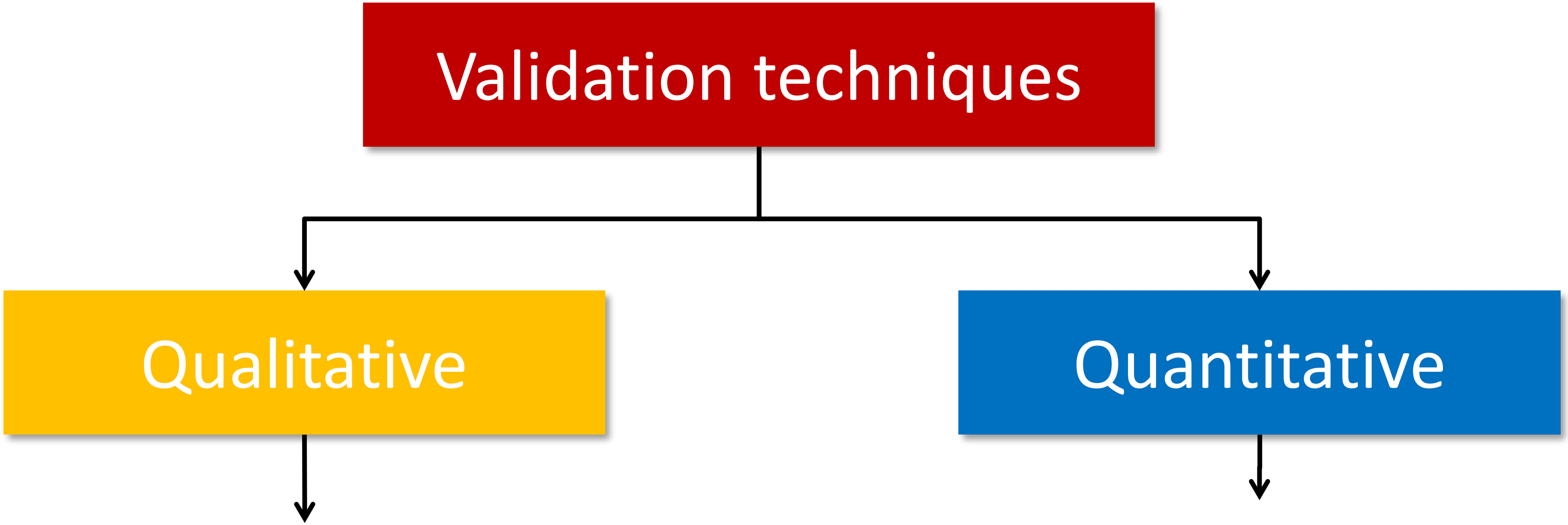
- Often negotiation-based
  - Delphi, facilitator + mediator techniques, etc.

Agenda

1. Requirements validation - overview

2. Requirements validation techniques

Requirements validation techniques



Validation techniques

Qualitative

- Inspection (review)
- Acceptance testing

Quantitative

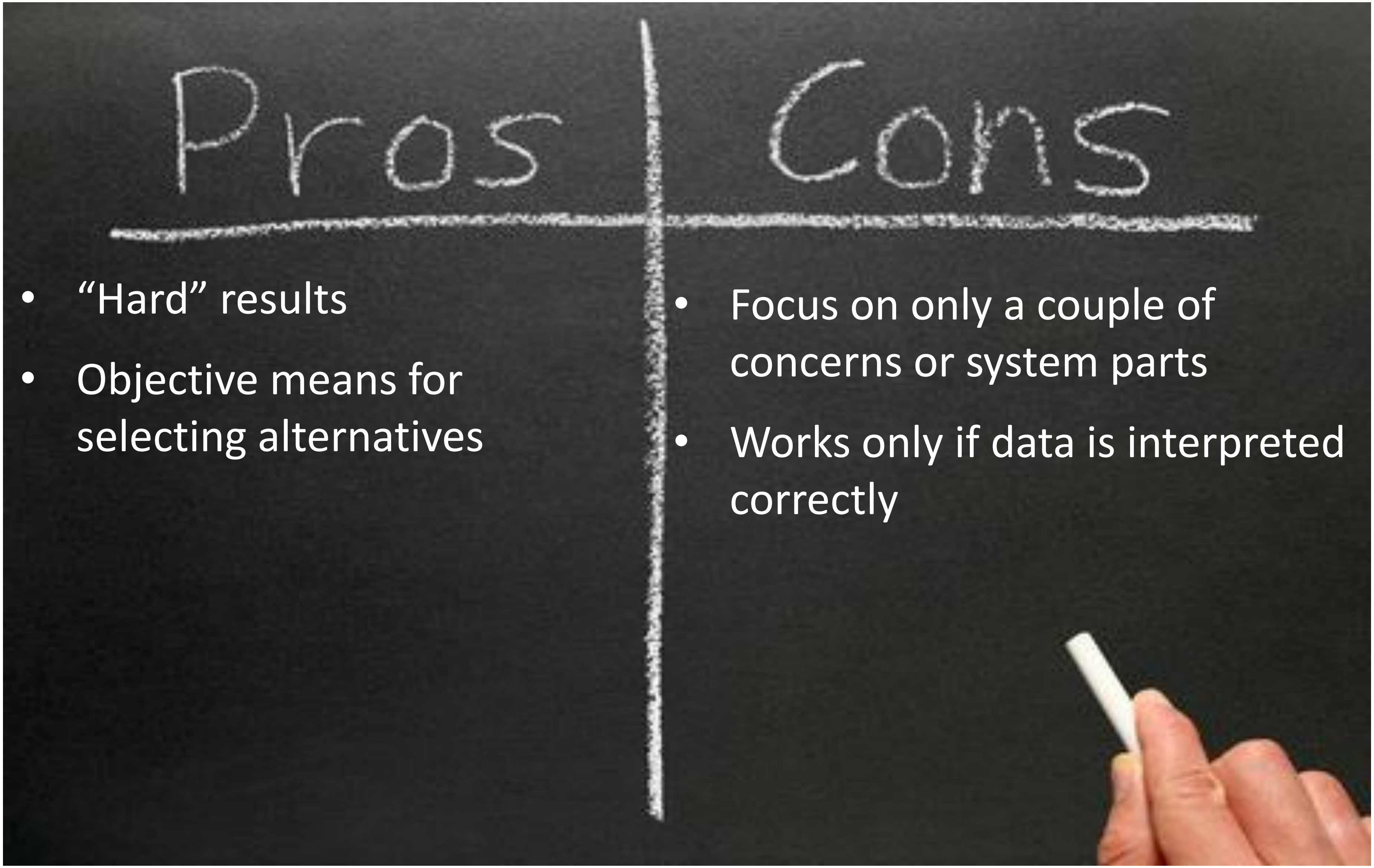
- Model checking
- Simulation
- Prototyping

Quantitative validation techniques

- Model checking (specification validation)
  - Requirement models verified using formal methods
  - Formal proof of critical properties

- Simulation and prototyping
  - Investigates dynamic system behavior
  - Simulation: specification validation
    - Simulate execution of specification, may visualize animated models
  - Prototyping: product validation
    - Partial implementation of requirements
    - Judge practical usefulness of specified system in real application context

Quantitative



- “Hard” results
- Objective means for selecting alternatives

- Focus on only a couple of concerns or system parts
- Works only if data is interpreted correctly

Inspection (review)

- Most common, simple and pragmatic method
  - Most evident errors can be detected
  - On average detect 60% to 80% of requirements errors

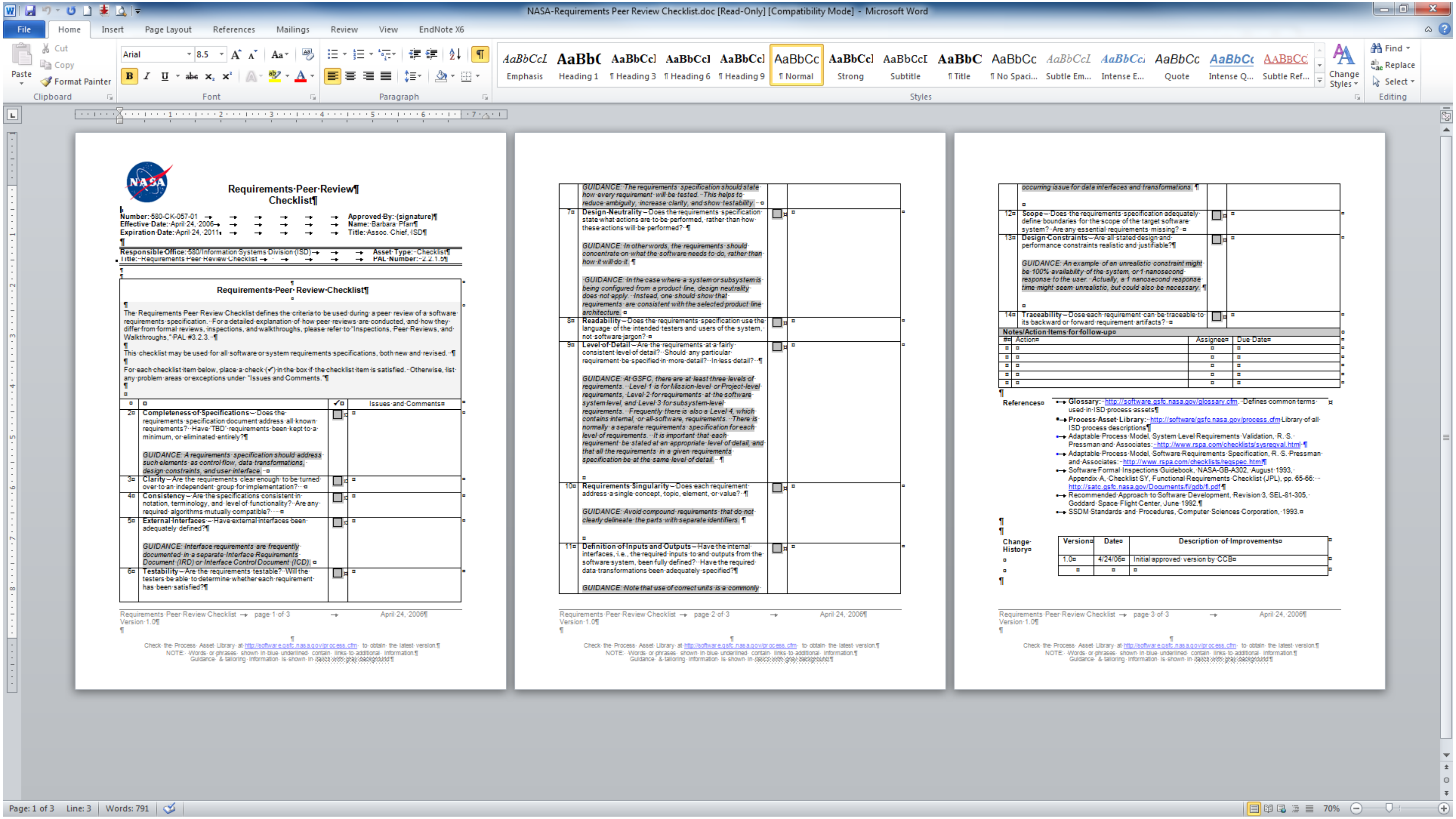
- Basic types
  - Walkthrough
    - Author guides experts through specification
    - Retrospective view on requirements
  - Expert review
    - Experts check specification
    - External and neutral view on requirements

Pre-inspection activities

- Define
    - Checking rules (checklist)
        - Guides and structures review process
        - Unambiguous, clear enough to test
    - Defect (violation of a rule)
        - E.g., if there are 10 ambiguous terms in one requirement: we have 10 defects
    - When to exit from inspection
        - Exit condition (e.g., maximum 1 major defect / page)

Example

- NASA software requirements specification review checklist templates



Requirements Peer Review Checklist

Number: 580-CK-057-01 Effective Date: April 24, 2006 Expiration Date: April 24, 2011

Approved By: (signature) Name: Barbara Pfarr Title: Assoc. Chief, ISD

Responsible Office: 580/Information Systems Division (ISD) Asset Type: Checklist

Title: Requirements Peer Review Checklist PAL Number: 2.2.1.5

Requirements Peer Review Checklist

The Requirements Peer Review Checklist defines the criteria to be used during a peer review of a software requirements specification. For a detailed explanation of how peer reviews are conducted, and how they differ from formal reviews, inspections, and walkthroughs, please refer to "Inspections, Peer Reviews, and Walkthroughs," PAL #3.2.3.

This checklist may be used for all software or system requirements specifications, both new and revised.

For each checklist item below, place a check (✓) in the box if the checklist item is satisfied. Otherwise, list any problem areas or exceptions under "Issues and Comments."

		✓	Issues and Comments
2	**Completeness of Specifications** – Does the requirements specification document address all known requirements? Have 'TBD' requirements been kept to a minimum, or eliminated entirely? *GUIDANCE: A requirements specification should address such elements as control flow, data transformations, design constraints, and user interface.*	☐	
3	**Clarity** – Are the requirements clear enough to be turned over to an independent group for implementation?	☐	
4	**Consistency** – Are the specifications consistent in notation, terminology, and level of functionality? Are any required algorithms mutually compatible?	☐	
5	**External Interfaces** – Have external interfaces been adequately defined? *GUIDANCE: Interface requirements are frequently documented in a separate Interface Requirements Document (IRD) or Interface Control Document (ICD).*	☐	
6	**Testability** – Are the requirements testable? Will the testers be able to determine whether each requirement has been satisfied? *GUIDANCE: The requirements specification should state how every requirement will be tested. This helps to reduce ambiguity, increase clarity, and show testability.*	☐	
7	**Design Neutrality** – Does the requirements specification state what actions are to be performed, rather than how these actions will be performed? *GUIDANCE: In other words, the requirements should concentrate on what the software needs to do, rather than how it will do it.* *GUIDANCE: In the case where a system or subsystem is being configured from a product line, design neutrality does not apply. Instead, one should show that requirements are consistent with the selected product line architecture.*	☐	
8	**Readability** – Does the requirements specification use the language of the intended testers and users of the system, not software jargon?	☐	
9	**Level of Detail** – Are the requirements at a fairly consistent level of detail? Should any particular requirement be specified in more detail? In less detail? *GUIDANCE: At GSFC, there are at least three levels of requirements. Level 1 is for Mission-level or Project-level requirements, Level 2 for requirements at the software system level, and Level 3 for subsystem-level requirements. Frequently there is also a Level 4, which contains internal, or all-software, requirements. There is normally a separate requirements specification for each level of requirements. It is important that each requirement be stated at an appropriate level of detail, and that all the requirements in a given requirements specification be at the same level of detail.*	☐	
10	**Requirements Singularity** – Does each requirement address a single concept, topic, element, or value? *GUIDANCE: Avoid compound requirements that do not clearly delineate the parts with separate identifiers.*	☐	
11	**Definition of Inputs and Outputs** – Have the internal interfaces, i.e., the required inputs to and outputs from the software system, been fully defined? Have the required data transformations been adequately specified? *GUIDANCE: Note that use of correct units is a commonly occurring issue for data interfaces and transformations.*	☐	
12	**Scope** – Does the requirements specification adequately define boundaries for the scope of the target software system? Are any essential requirements missing?	☐	
13	**Design Constraints** – Are all stated design and performance constraints realistic and justifiable? *GUIDANCE: An example of an unrealistic constraint might be 100% availability of the system, or 1 nanosecond response to the user. Actually, a 1 nanosecond response time might seem unrealistic, but could also be necessary.*	☐	
14	**Traceability** – Dose each requirement can be traceable to its backward or forward requirement artifacts?	☐	

Notes/Action Items for follow-up

#	Action	Assignee	Due Date

References

- **Glossary:** http://software.gsfc.nasa.gov/glossary.cfm. Defines common terms used in ISD process assets
- **Process Asset Library:** http://software/gsfc.nasa.gov/process.cfm Library of all ISD process descriptions
- Adaptable Process Model, System Level Requirements Validation, R. S. Pressman and Associates: http://www.rspa.com/checklists/sysreqval.html
- Adaptable Process Model, Software Requirements Specification, R. S. Pressman and Associates: http://www.rspa.com/checklists/reqspec.html
- Software Formal Inspections Guidebook, NASA-GB-A302, August 1993, Appendix A, Checklist SY, Functional Requirements Checklist (JPL), pp. 65-66: http://satc.gsfc.nasa.gov/Documents/fi/gdb/fi.pdf
- Recommended Approach to Software Development, Revision 3, SEL-81-305, Goddard Space Flight Center, June 1992.
- SSDM Standards and Procedures, Computer Sciences Corporation, 1993.

Change History

Version	Date	Description of Improvements
1.0	4/24/06	Initial approved version by CCB

Requirements Peer Review Checklist page 1 of 3 April 24, 2006
Version 1.0

Check the Process Asset Library at http://software.gsfc.nasa.gov/process.cfm to obtain the latest version.
NOTE: Words or phrases shown in blue underlined contain links to additional information.
Guidance & tailoring information is shown in italics with gray background

Requirements Peer Review Checklist page 2 of 3 April 24, 2006
Version 1.0

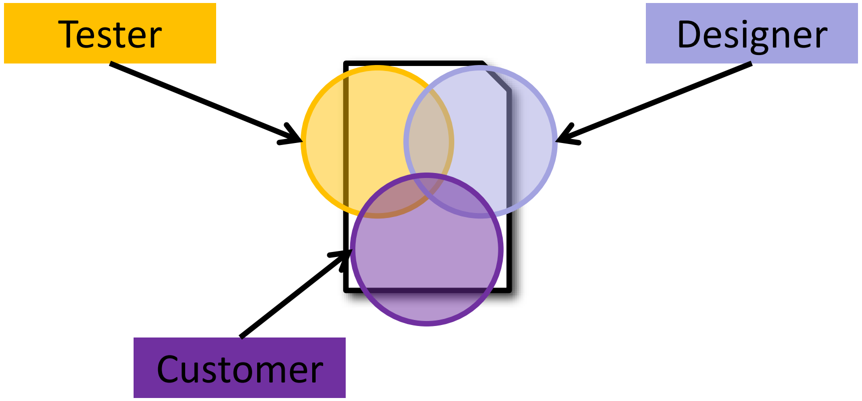
Check the Process Asset Library at http://software.gsfc.nasa.gov/process.cfm to obtain the latest version.
NOTE: Words or phrases shown in blue underlined contain links to additional information.
Guidance & tailoring information is shown in italics with gray background

Requirements Peer Review Checklist page 3 of 3 April 24, 2006
Version 1.0

Check the Process Asset Library at http://software.gsfc.nasa.gov/process.cfm to obtain the latest version.
NOTE: Words or phrases shown in blue underlined contain links to additional information.
Guidance & tailoring information is shown in italics with gray background

Perspective-based reading (PBR)

- Read and analyze requirements from different perspectives
  - E.g., end user, tester, architect, maintainer



- Why
  - Inspectors often do not know how to read the document
  - Different perspectives: requirements more reliability reflect actual needs
  - Developers learn how to write document but not how to read them

B. Regnell et al.: *Are the Perspectives really Different? – Further Experimentation on scenario-based reading of Requirements.* Empirical Software Engineering (2012)

Steps

- Select perspectives for reading
  - E.g.,
    - Designer: uses requirements to produce a system design
    - Tester: produces a test plan to ensure that the system can meet requirements
    - Customer: wants to make sure requirements have been adequately captured
- Define procedure (instructions / scenario) for reading
  - Can include checklist (questions) for reading
- Conduct review
  - Include paraphrasing: explain requirements in the reviewer’s own words
- Provide feedback

Requirements and acceptance testing

Self-study

- RE and acceptance testing are naturally intertwined
  - Acceptance: process of assessing whether system satisfies requirements
  - Acceptance test: test that assesses whether system satisfies requirements

- When writing requirements
  - Can acceptance tests be written to validate them?
    - Mentally execute acceptance test cases

- Process of writing ATs helps identify ambiguous requirements

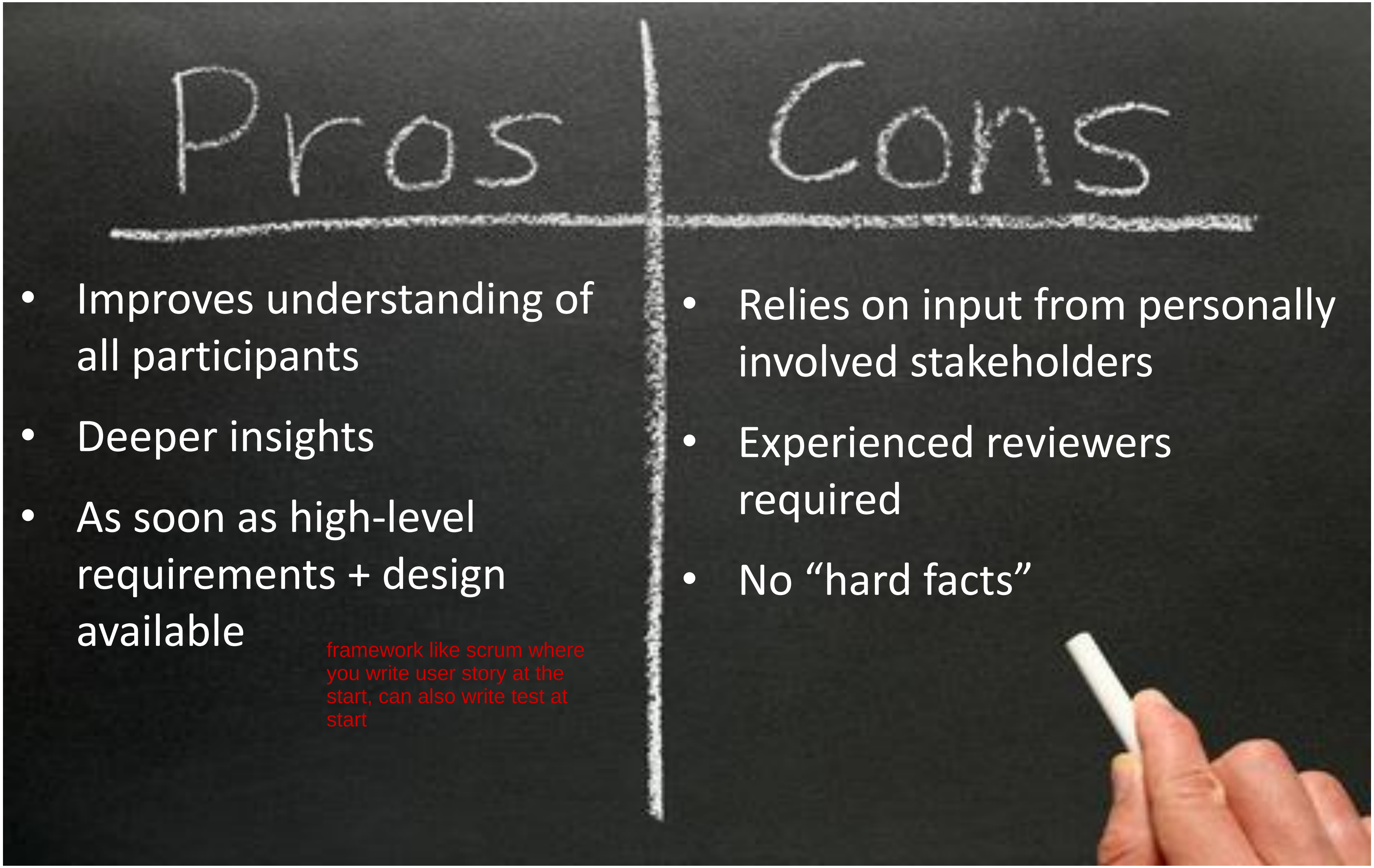
Choosing acceptance test cases

Self-study

- 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 and important

Qualitative



- Improves understanding of all participants
- Deeper insights
- As soon as high-level requirements + design available

framework like scrum where you write user story at the start, can also write test at start

- Relies on input from personally involved stakeholders
- Experienced reviewers required
- No “hard facts”

Review costs – thought experiment (1)

- Assumption
  - Random requirements specification: 100 pages
  - 2 hours for checking two random pages (page a, page b)
  - Total time: 100 hours

- Count all major defects

- Findings
  - Page a: 40 defects; page b: 30
  - Average: 35 defects / page * 100 pages = 3,500 defects in total

Review costs – thought experiment (2)

- More assumptions
  - Defect has 1/3 chance of causing loss
  - Each loss requires on average 10 person-hours to correct
  - Rework cost is 1/3 *3,500 * 10 = 11667 person-hours for rework

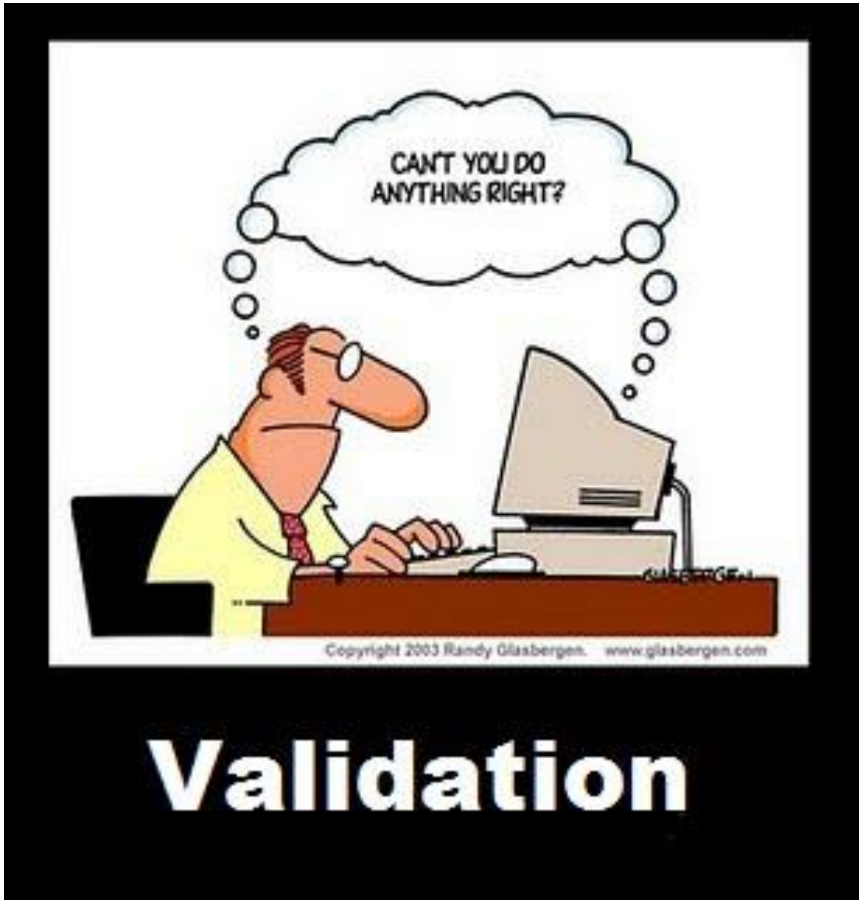
- Final comparison
  - 100 hours for inspection
  - 11667 hours for rework

Summary

1. Requirements validation – overview ✓

2. Requirements validation techniques ✓

Validation


CAN'T YOU DO ANYTHING RIGHT?
Copyright 2003 Randy Glasbergen. www.glasbergen.com
Validation