Lecture 6: Requirements Validation and Verification

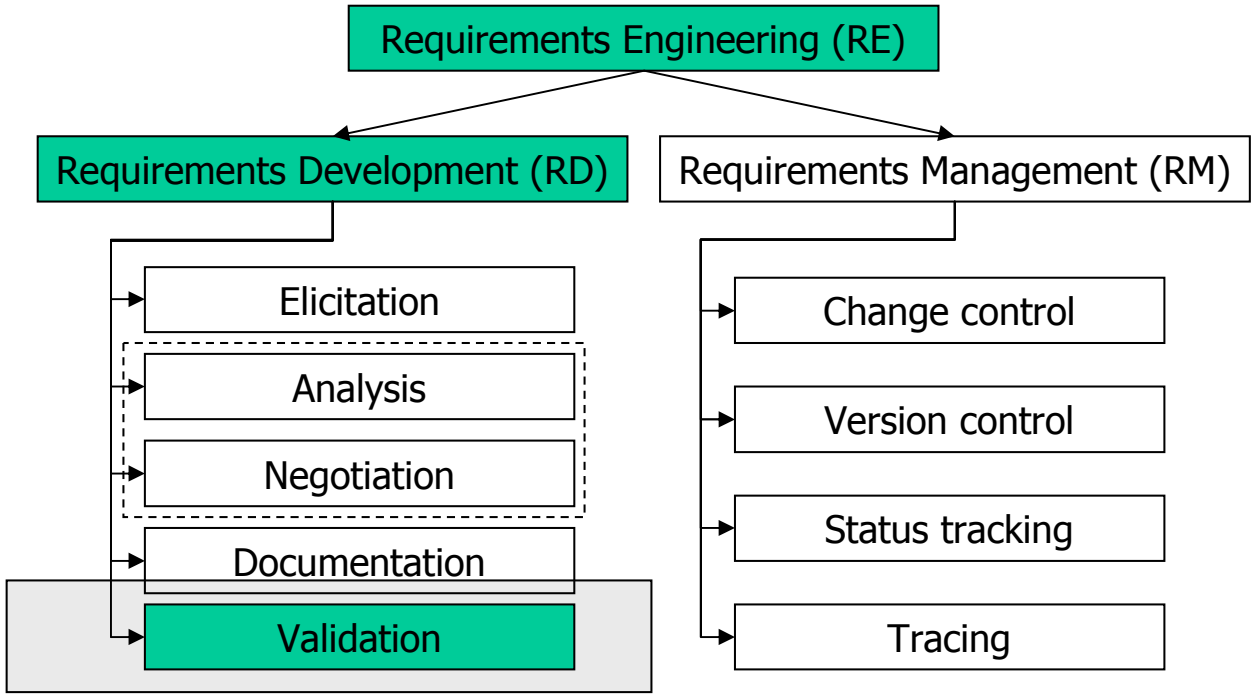
Requirements Management and Systems Engineering (ITKS451), Autumn 2008

Artem Katasonov
University of Jyväskylä

In the previous episode:

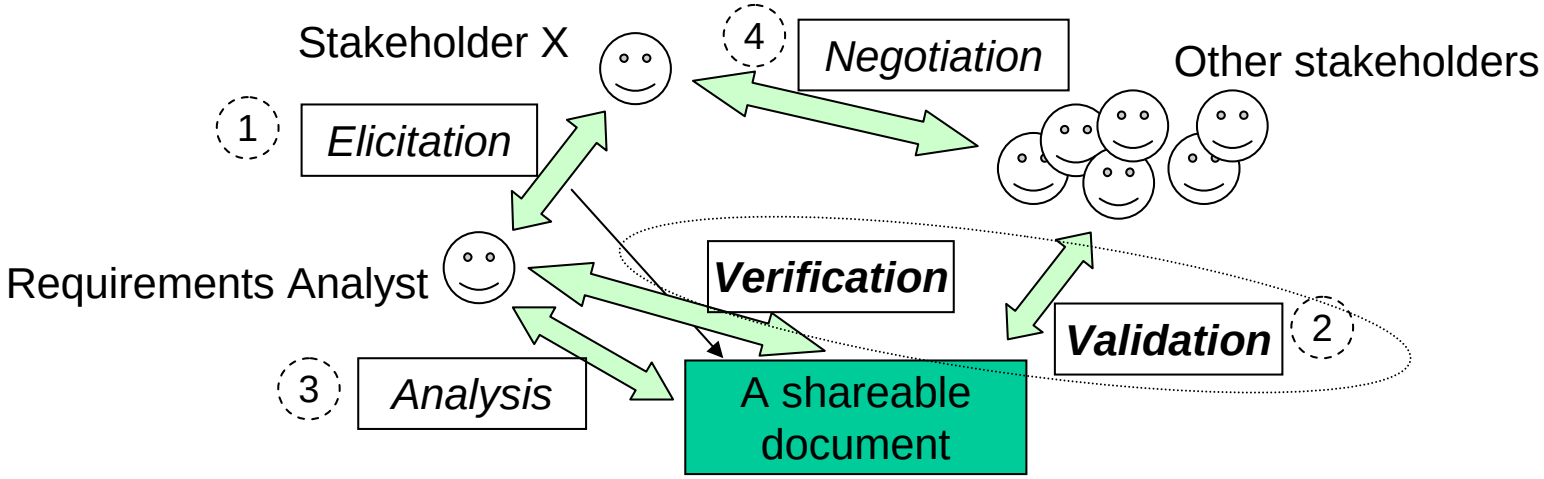
- Requirements elicitation is a collaborative process, in which a system stakeholder and a requirements analyst attempt to construct a proposition of a solution to those problems of that stakeholder, which are to be solved by the developed system.
- For elicitation from stakeholders of type “users”, use case approach is good.
- Remember, however, that users are not the only stakeholders to consider, and that use cases can be neither the first nor the last step of requirements development.
- Remember also to consider different user classes, analyze alternative courses and exceptions for a use case, and consider possible interferences and dependencies among use cases.

Requirements validation and verification



- *Validation (& verification)* - the process of checking whether the requirements, as identified, do not contradict the expectations about the system of various stakeholders, and do not contradict each other.
- It is Requirements Quality Control.

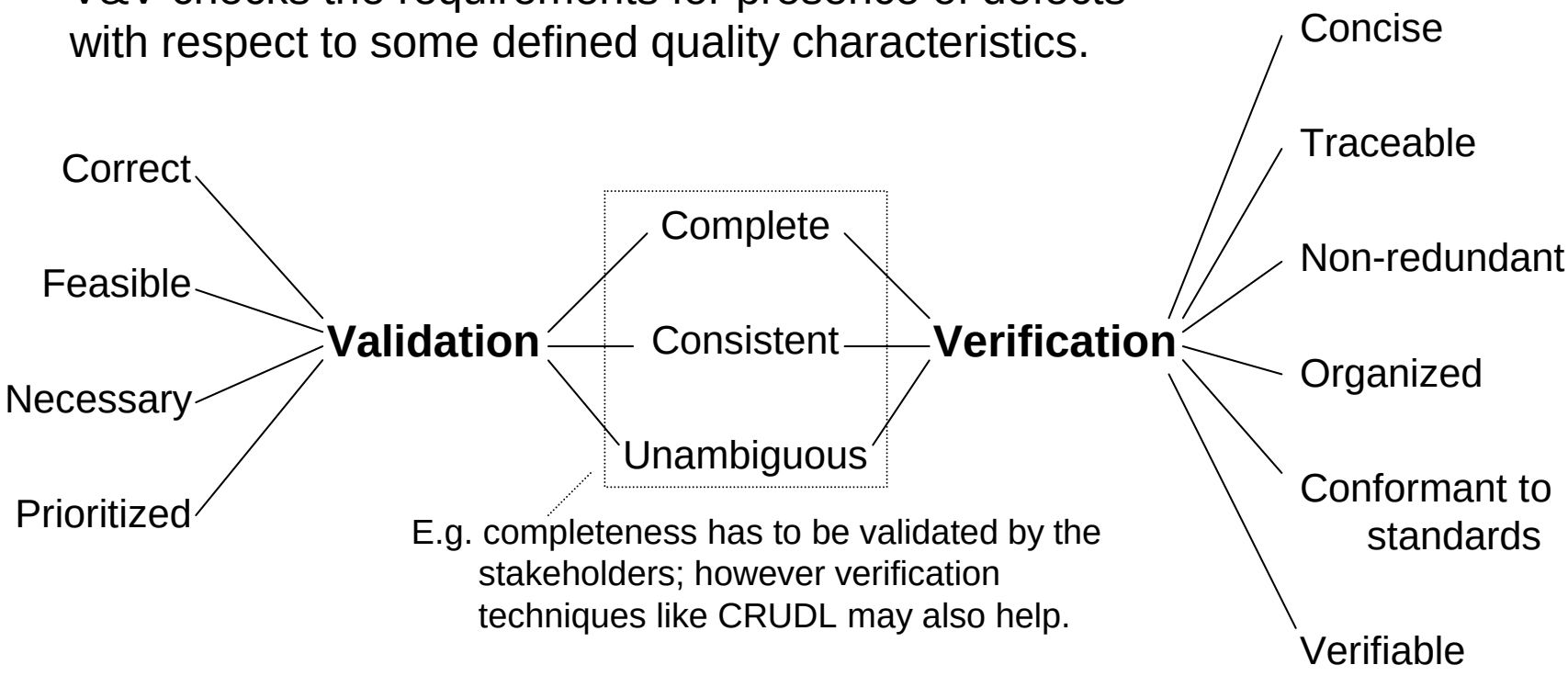
Validation vs. verification



- *Validation* – "Am I building the right *product*?" Checking a work product against higher-level work products or authorities that frame *this particular* product.
  - Requirements are validated by stakeholders.
- *Verification* – "Am I *building* the product right?" Checking a work product against some standards and conditions imposed on this *type* of product and the *process* of its development.
  - Requirements are verified by the analysts mainly.

Validation vs. verification (2)

- V&V checks the requirements for presence of defects with respect to some defined quality characteristics.



- Henceforth, we will draw no distinction and call both “validation”.

Validation in literature

- Literature tends to discuss requirements validation as a a heterogeneous process based on application of a great variety of independent techniques.
- Most books present validation with a bulleted list of “good practices”.
- In our own research, we attempted to develop a more coherent view. The result is the article *Requirements quality control: a unifying framework* just published in the Requirements Engineering Journal, 11(1), 2006 http://www.springerlink.com/openurl.asp?genre=article&id=doi:10.1007/s00766-005-0018-1
- The rest of this lecture follows our generalizing perspective.

Goal of validation

- Two key assumptions that frame traditional RE:
  1. requirements exist ‘out there’ in the minds of stakeholders, and they only need to be elicited through various mechanisms.
  2. the key stakeholders operate in a state of goal congruence, in which there is widespread agreement on the general goals of the system under development.

- Under these assumptions, validation of requirements is nothing more than checking whether the analysts have understood the stakeholders’ intention correctly and have not introduced any errors when writing the specification.

- The above assumptions seldom hold true
  - Requirements are not discovered but constructed
  - There is usually at least some goal incongruence between stakeholders

Goal of validation (2)

- Validation is not a mechanical process of checking documents.
- It is more *an issue of communicating requirements, as constructed by the analysts, back to the stakeholders whose goals those requirements are supposed to meet, and to all those other stakeholders, with whose goals those requirements may conflict.*
- It is an information feedback link needed to:
  - give the stakeholders a chance to check early whether the solution proposed will really solve their problem
  - stimulate the evolution of customers' understanding (of what is possible) and therefore act also as a catalyst of the elicitation process

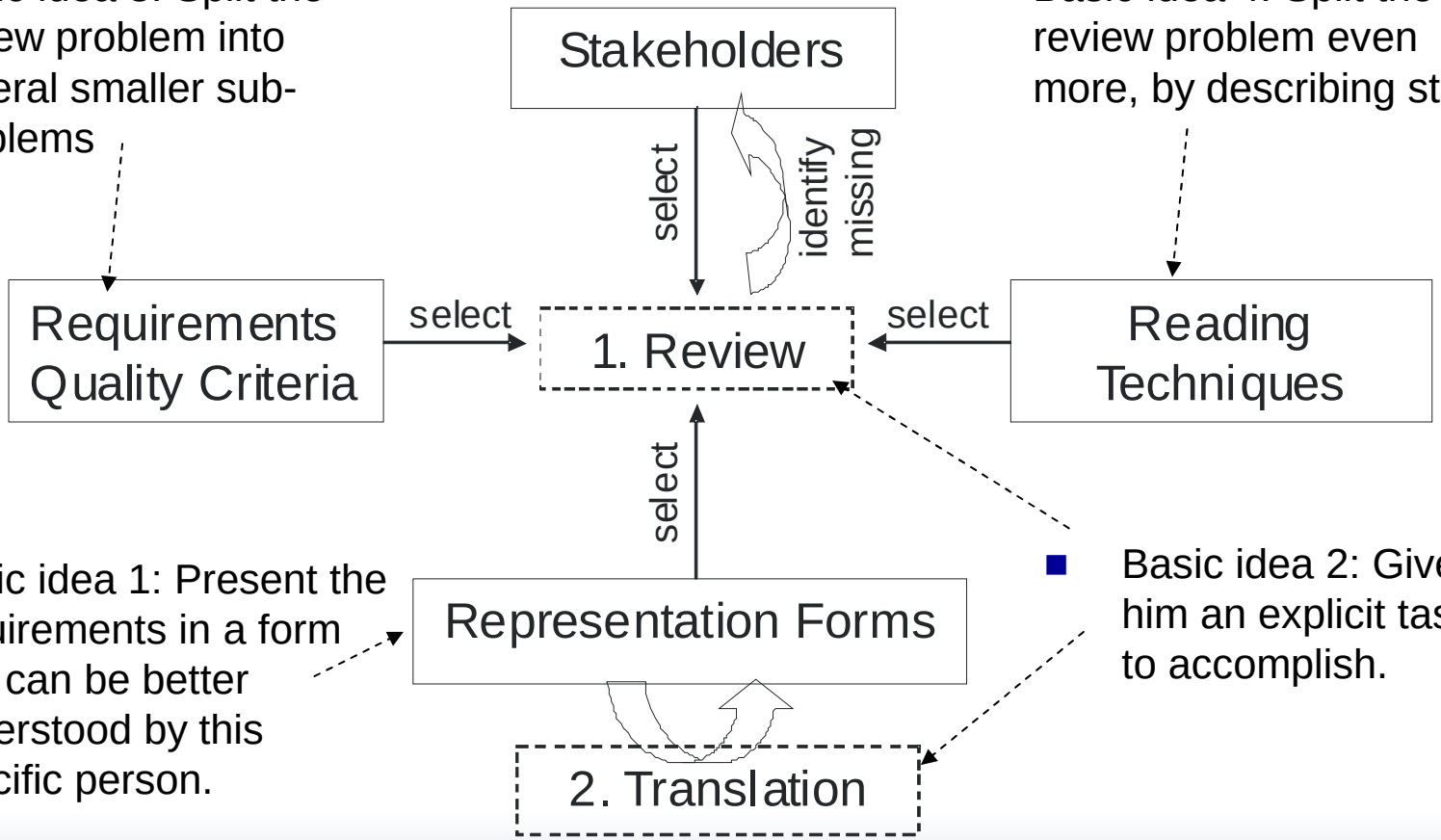
Main issue in validation

- While elicitation and analysis are attempts to cross the boundary from the domain world to the machine world, validation is an attempt to cross the same boundary in the opposite direction.
- The difficulty of the former is well acknowledged, and the latter can be almost as difficult.
- Main problem is achieving a sufficient level of understanding of the stated requirements by a particular stakeholder, which may be hindered by:
  - lack of technical expertise
  - lack of some special domain knowledge (domain of another stakeholder)

Validation basic ideas and framework

- Basic idea 3: Split the review problem into several smaller sub-problems

- Basic idea 4: Split the review problem even more, by describing steps



- Basic idea 1: Present the requirements in a form that can be better understood by this specific person.

- Basic idea 2: Give him an explicit task to accomplish.

Validation framework - Activities

- **Review** - rely on human intelligence in hope that people might find some problems with requirements.
  - Also, if the requirements document or some part of it is in a sufficiently formal language, some properties can be automatically checked by software.

- **Translation** - translate requirements to an alternative form. This has a double advantage:
  1. The translation process may fail (rewriting turns out to be partially impossible), thereby pointing to some ambiguity, completeness, consistency, etc. defects. Also, if two or more persons accomplish rewriting independently and the results are significantly different, this also points to problems (mainly ambiguity) in the document.
  2. After rewriting, requirements in the new form are again reviewed. The idea is that this form is believed to be better understandable to a specific group of reviewers. Also, the rewritten form may be amenable to automated checking.

Validation framework - Pools

- **Stakeholders**. Treated earlier.
- **Requirements Quality Criteria**. Treated earlier.
- **Representation Forms**. Requirements can be represented in a variety of different forms.
- **Reading Techniques**. There are different techniques to be used during a review, mainly related to the type of guidance the reviewers receive for accomplishing their task.

- Therefore, a *requirements review* is a process in which a subset of the system stakeholders investigates the requirements represented in one or more of the available forms, using one or more of the available reading techniques, based on a subset of the quality criteria defined.

Reviewing requirements

- Every time someone other than the author of a software work product examines the product for problems, a technical *review* is taking place (Wiegers, 2003).
- Requirements reviewing is the basic activity in which they are validated, or in which requirements defects are identified.

- Reviews can be informal or formal.
- *Informal* reviews:
  - Distribution of the work product to several peers to look through.
  - Walk-through in which the author describes the product in front of an audience and solicits comments.
  - Giving the product to a person to examine, and watching him doing that. “Looking for the furrowed brow that indicates a puzzled user”.
- The best-known type of *formal* review is called ***inspection***.

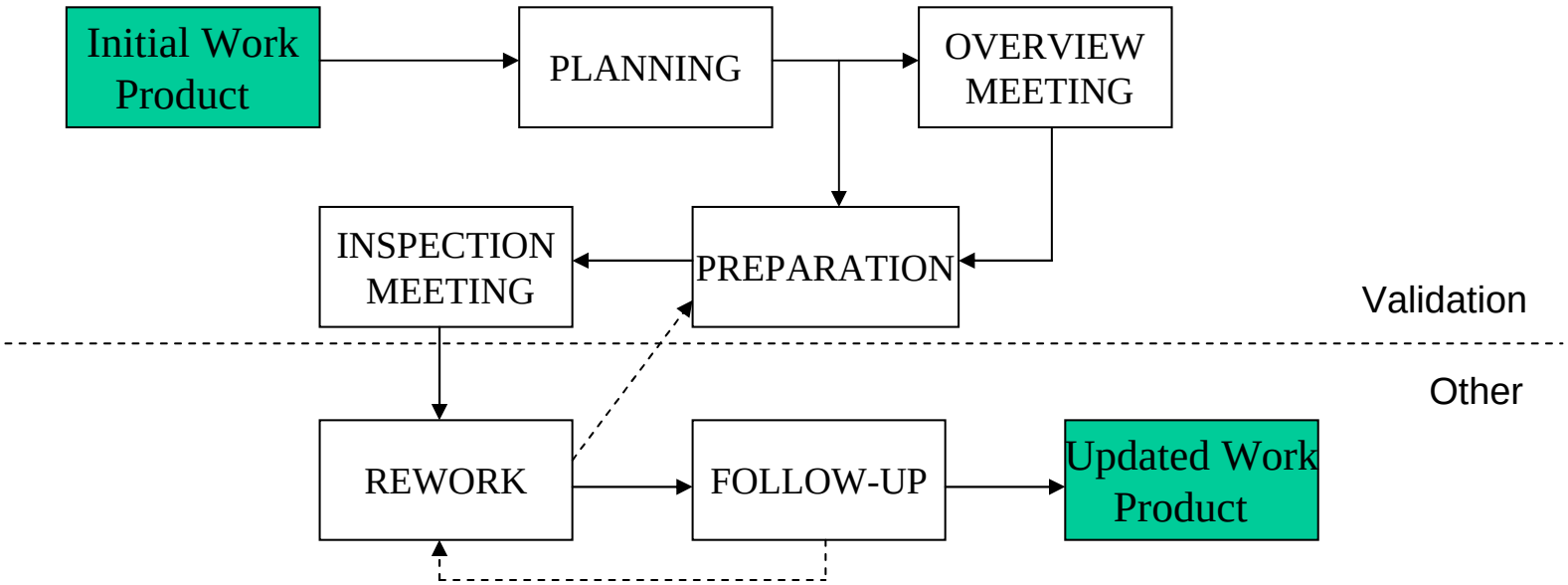
Inspection

- Requirements inspection involves creating a team of inspectors that should represent different perspectives.

- In general, the team should include the author of the inspected document, the author of any predecessor document or specification, and people who have to do work based on the document (Wiegers, 1999).
- Therefore, a requirements inspection team should include requirements engineers (authors), design engineers (will do work based on requirements), and various other stakeholders (sources of requirements).

- Inspection is a formal review because it proceeds in a sequence of well-defined stages.

Inspection (2)



- Important:
  - Each inspector works with the inspected document on his own, during "preparation" phase. It is the phase when most of problems are identified.
  - During the inspection meeting, inspectors discuss identified issues, and probably find some additional.

Reading Techniques

- In both formal and informal reviews, an important factor is the level of guidance the reviewers receive for accomplishing their task, or *reading technique*.

- Reading techniques:
  - Ad-hoc,
  - Checklist-based,
  - Defect-based,
  - Perspective-based,
  - Scenario-based,
  - Pattern-based.

Ad-hoc reading and checklists

- *Ad-hoc reading* – no guidance is provided, reviewers use only their own knowledge and experience to identify defects.

- *Checklist-based reading* – a list of questions is provided specifying what properties of the document must be checked and what specific problems should be searched for.
  - Usually the only alternative that is discussed in requirements engineering books.
  - Every relevant requirements quality criterion is rewritten in the form of a question, or refined into two or more questions.
  - A checklist works just as a reminder for reviewers of those quality criteria.

Checklist example (Wiegers, 2003)

1. Organization and Completeness

- Are all internal cross-references to other requirements correct?
- Are all requirements written at a consistent and appropriate level of detail?
- Do the requirements provide an adequate basis for design?
- Is the implementation priority of each requirement included?
- Are all external hardware, software, and communication interfaces defined?
- Have algorithms intrinsic to the functional requirements been defined?
- Does the SRS include all of the known customer or system needs?
- Is any necessary information missing from a requirement? If so, is it identified as TBD?
- Is the expected behavior documented for all anticipated error conditions?

2. Correctness

- Do any requirements conflict with or duplicate other requirements?
- Is each requirement written in clear, concise, unambiguous language?
- Is each requirement verifiable by testing, demonstration, review, or analysis?
- Is each requirement in scope for the project?
- Is each requirement free from content and grammatical errors?
- Can all of the requirements be implemented within known constraints?
- Are any specified error messages unique and meaningful?

Checklist example (Wiegers, 2003) (2)

3. Quality Attributes

- Are all performance objectives properly specified?
- Are all security and safety considerations properly specified?
- Are other pertinent quality attribute goals explicitly documented and quantified, with the acceptable tradeoffs specified?

4. Traceability

- Is each requirement uniquely and correctly identified?
- Can each software functional requirement be traced to a higher-level requirement (e.g., system requirement, use case)?

5. Special Issues

- Are all requirements actually requirements, not design or implementation solutions?
- Are the time-critical functions identified, and timing criteria specified for them?
- Are all significant consumers of scarce resources (memory, network bandwidth, processor capacity, etc.) identified, and is their anticipated resource consumption specified?
- Have internationalization issues been adequately addressed?

Reading following a procedure

- Not only a list of review questions but also a collection of procedures to follow is provided, which describe the steps to be accomplished in order to answer the questions.

- *Defect-based reading* – each procedure is aiming for detecting a particular type of defects, different procedures are usually assigned to different reviewers.
  - Some empirical evidence exists that this may outperform checklist-based and ad-hoc approaches.

- *Perspective-based reading* – each procedure is based on the viewpoint of a particular stakeholder.

Defect-based procedure example

1. Identify all data objects mentioned in the overview (e.g., hardware component, application variable, abbreviated term or function):
   a) Are all data objects mentioned in the overview listed in the external interface section?
2. For each data object appearing in the external interface section determine the following information:
   - Object name:
   - Class: (e.g., input port, output port, application variable, abbreviated term, function)
   - Data type: (e.g., integer, time, Boolean, enumeration)
   - Acceptable values: Are there any constraints, ranges, limits for the values of this object?
   - Failure value: Does the object have a special failure value?
   - Units or rates:
   - Initial value:

   a) Is the object's specification consistent with its description in the overview?
   b) If object represents a physical quantity, are its units properly specified?
   c) If the object's value is computed, can that computation generate a non-acceptable value?
3. For each functional requirement identify all data object references:
   a) Do all data object references obey formatting conventions?
   b) Are all data objects referenced in this requirement listed in the input or output sections?
   c) Can any data object use be inconsistent with the data object's type, acceptable values, failure value, etc.?
   d) Can any data object definition be inconsistent with the data object's type, acceptable values, failure value, etc.?

- A procedure for detecting data-related requirements defects

Porter, A. A., Votta, L. G. J., and Basili, V. R. (1995) *Comparing detection methods for software requirements inspections: a replicated experiment*. IEEE Transactions on Software Engineering 21(6), 563–575

Scenario-based reading

- *Scenario-based reading* - a set of concrete scenarios is provided. Reviewers walk through the sequence of events in each scenario and examine the requirements document for presence, correctness, etc. of requirement statements that cover those.
  - May be usage scenarios, maintenance work scenarios, etc.
  - "Scenario-based validation" is often mentioned in books and other requirements literature as a separate validation technique.
- McGregor and Sykes "guided inspection":
  - Inspectors prepare scenarios, then
  - The authors of the reviewed document then explain how the system described by the document is assumed to handle these cases, while
  - Inspectors follow explanations and look for problems:
    - Inappropriate system's behavior,
    - Information has left in the head of the author and was not written down.
  - This is an interesting variant of scenario-based reading, which could also be called "requirements desk testing".

McGregor, J. D. and Sykes, D. A. (2001): *A Practical Guide to Testing Object-Oriented Software*. Addison-Wesley.

Pattern-based reading

- *Pattern-based reading* - a set of patterns is provided to reviewers that they can use when validating requirements against scenarios.
  - A set of requirements patterns is to be defined. Examples:
    - 'MACHINE-FUNCTION' pattern – a good requirements document shall include at least one functional requirement statement for each action in which the software system is involved (essential facts about requirement documents).
    - 'COLLECT-FIRST-OBJECTIVE-LAST' pattern – a good mechanical device with which a person interacts using one or more personal items must ensure that the person will not go away leaving those personal items at the device. (ideas essential to design of systems).
  - Each requirements pattern can be represented by a formal *validation frame* that describes what a part of a scenario should look like to make the pattern applicable, and what should then be searched for in the requirements document.

Maiden, N., Cisse, M., Perez, H., and Manuel, D. (1998) *CREWS validation frames: Patterns for validating systems requirements*. Proc. 4th REFSQ International Workshop on Requirements Engineering: Foundation for Software Quality

Translating requirements to alternative forms

- In practice, requirements are usually documented in structured natural language.
  - All natural languages are inherently ambiguous.
  - The requirements are often written in English although many stakeholders are not fluent in it.
  - Presenting the requirements as a collection of separate statements is probably not the best understandable form for either customers or developers.

- Three important observations:
  - There are also alternative forms for representing requirements and communicating them to stakeholders (although they usually allow only partial view of requirements).
  - A particular form may be understandable better than natural language description for a particular stakeholder.
  - When translating requirements from one form to another, this activity may succeed only if requirements are sufficiently complete, consistent, etc.

Translating requirements (2)

- It is not considered feasible to maintain several concurrent representations of requirements because of modifiability problems.
- However, for validating them, rewriting natural language into other forms is considered very useful.
- Alternative forms for representing and communicating requirements:
  - User manual,
  - Visualizations, e.g. diagrams,
  - Lightweight formal models,
  - Prototypes,
  - Test cases.

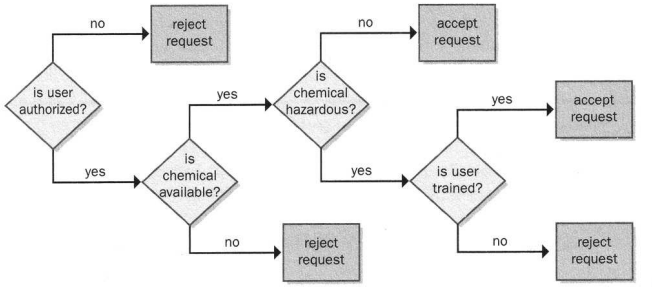
User manual

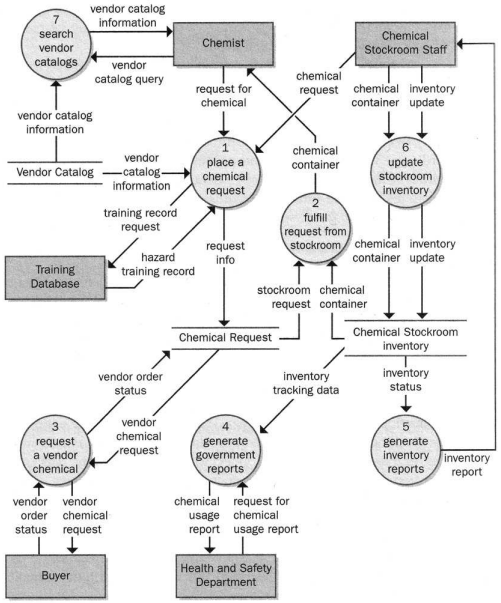
- A good manual describes all user-visible functionality in easily understandable language.
- It is still a natural language document, but with all ‘shall’ statements rewritten as if they were already implemented. It presents requirements in a more tangible and more coherent form than an SRS.
- However, a user manual is obviously only a partial view of requirements since it presents only functionality (not performance or other characteristics) and only that functionality visible to end users of the software.

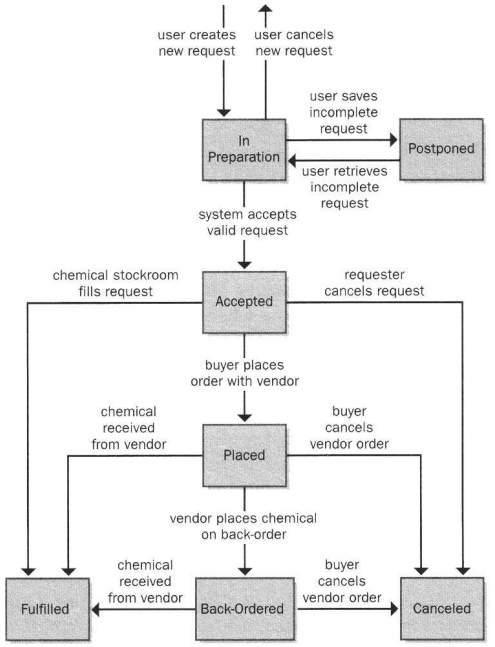
- A quality SRS should provide enough information needed for drafting the user manual. If the writer finds this (partially) impossible, this points to some problems with the SRS that must be investigated.
- After the draft is finished, it is to be reviewed by targeted stakeholders. Review can be informal or formal, be ad-hoc or use checklists, procedures or scenarios of type “suppose you want to do..”.

Visualizations

- Visualization is often seen as a way to help people gain insight from large and complex data sets.
- Graphical presentations link individual statements together and present a coherent picture of a slice of the system.
- Decision trees, data flow diagrams, state transition diagrams, etc. (will be discussed more in Lecture 7).







Lightweight formal models

- Over years, scientific community had developed a set of so-called *formal models*, including, e.g. Vienna Development Method (VDM), Z, Software Cost Reduction (SCR) – expressing requirements in formal logic and mathematically.
  - They have not been widely accepted in RE practice.
- More recently, it was proposed to use them in *lightweight* way. “Lightweight” indicates that the methods can be used for partial analysis on partial specifications, without a commitment to developing and baselining a complete, consistent formal specification.
  - I.e. some critical parts of a requirements specification are translated into a formal model for e.g. validation needs.
- Sufficient empirical evidence exists that just the attempt to translate requirements into a formal model helps to reveal a lot of defects.
- After translating, some properties of the partial formal specification can be automatically verified by software.

Prototypes

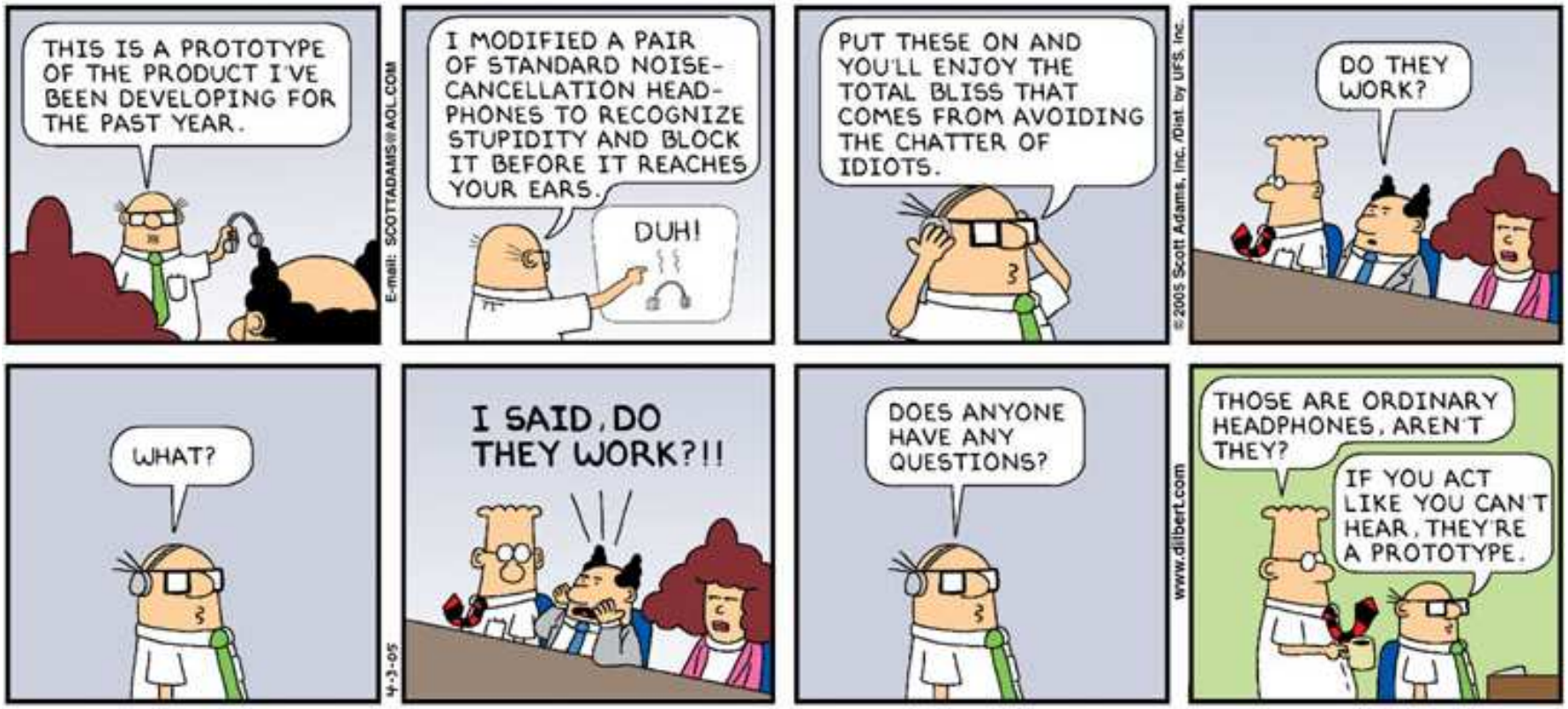
- A prototype makes requirements tangible.
- Users, managers, and other non-technical stakeholders usually find it very difficult to visualize how a written statement of requirements will translate into an executable software system.
- If a prototype is developed to demonstrate requirements, they find it easier to discover problems and suggest how the requirements may be improved.

- Simplest prototypes:
  - paper prototype - no executable software,
  - 'Wizard of Oz' prototype – only interface implemented, a person explains effects.

- Even paper prototypes help in discovering many requirements defects.

"The focus group destroyed the computers and burned down the computer room. We now know how they feel about pop-ups."

Wizard of Oz prototypes



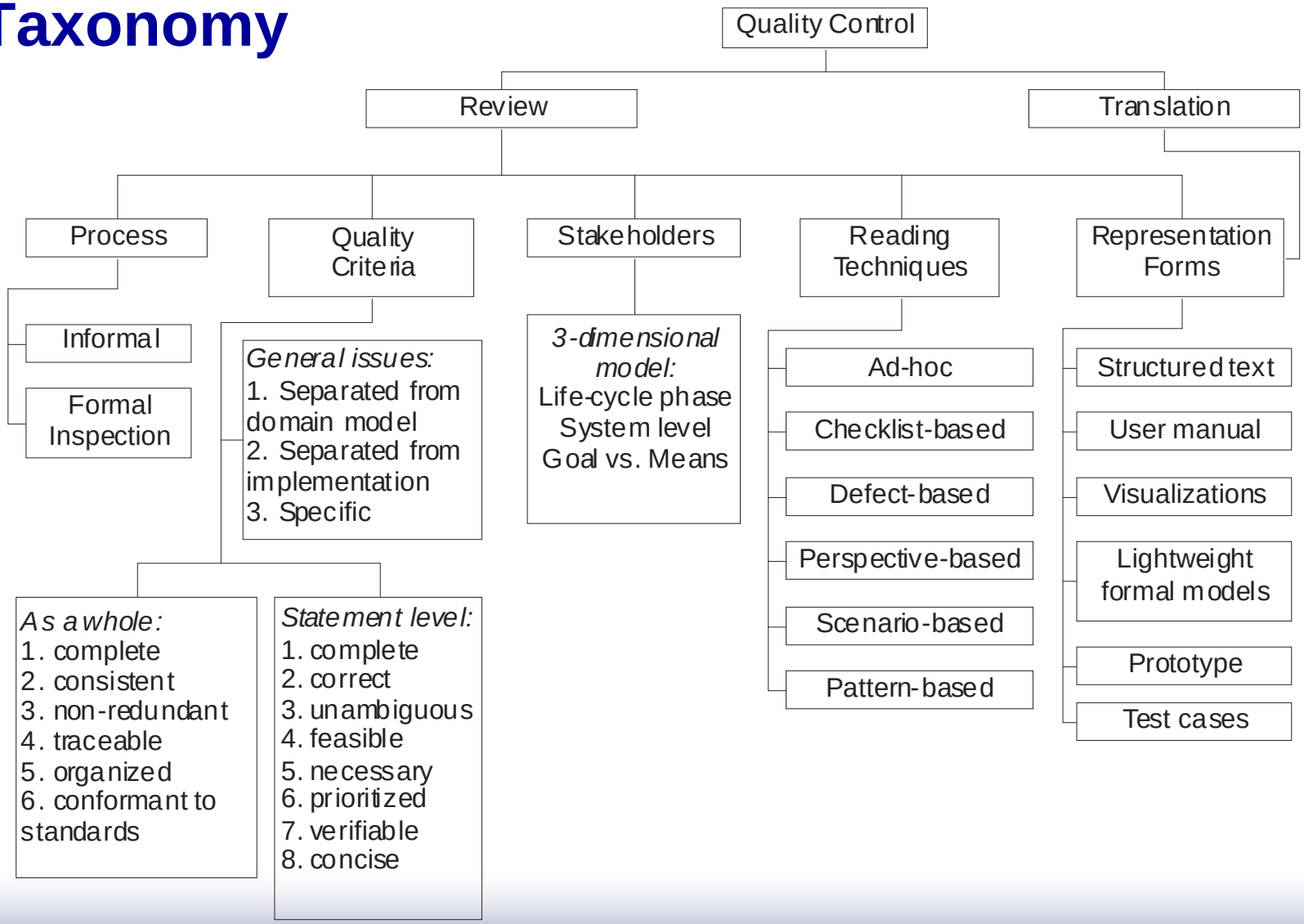
Test cases

- A desirable attribute of every requirement statement is that it should be verifiable, i.e. it should be possible to define one or more test cases that can check whether the requirement has been met.
- Even though the tests will be applied to the system only after implementation, designing tests at an early stage is an effective way of revealing requirements defects.
- A test case usually tests several related requirements, both functional and non-functional. Therefore, test cases may make the expected system behaviors clearer to all the stakeholders.
- Test cases designed for validation of requirements may also be only theoretical, e.g. be too expensive to run in practice.

Validation procedure

1. For each stakeholder, identify the *subset of requirements* that he should and can validate and/or verify.
2. For each stakeholder, select a *set of representation forms* that are believed to be best understandable for him, and can cover the set of requirements identified in step 1.
3. For each stakeholder, identify the *set of quality attributes* he should and can evaluate.
4. For each pair stakeholder - representation form, select the most appropriate and feasible *reading technique*.
5. Develop or reuse needed reading assistance tools, e.g. checklists, scenarios, for techniques identified in step 4.
6. Select stakeholders to perform translations to each needed representation form.
7. Commit translations.
8. Commit reviews.
9. If the defined end criteria are not fulfilled, fix the defects found and return to a step 1... 8 (depending on the scope of the change).

Taxonomy


Quality Control
Review
Translation
Process
Quality Criteria
Stakeholders
Reading Techniques
Representation Forms
Informal
Formal Inspection
General issues:
1. Separated from domain model
2. Separated from implementation
3. Specific
3-dimensional model:
Life-cycle phase
System level
Goal vs. Means
Ad-hoc
Checklist-based
Defect-based
Perspective-based
Scenario-based
Pattern-based
Structured text
User manual
Visualizations
Lightweight formal models
Prototype
Test cases
As a whole:
1. complete
2. consistent
3. non-redundant
4. traceable
5. organized
6. conformant to standards
Statement level:
1. complete
2. correct
3. unambiguous
4. feasible
5. necessary
6. prioritized
7. verifiable
8. concise

Lecture 6: Main points

- Requirements validation is a feedback link. It communicates requirements, as constructed by the analysts, back to the stakeholders whose goals those requirements are supposed to meet, and to all those other stakeholders, with whose goals those requirements may conflict.
- While literature mentions more than a dozen different requirements validation techniques, all of them can be generalized into two – reviewing requirements (represented in different forms) and translating them from a form to another.
- Reviews may be informal or formal, and may use different reading techniques.
- Some known forms for representing and communicating requirements are structured text (SRS), user manual, prototypes, diagrams, formal models, and test cases.
- A particular form may be better understandable for some group of stakeholders. Also, just the attempt of translating usually reveals some defects.

Marginally related..

- I have bought an active pillow

- Does not look very active. Is it broken?

- Let’s check if it is possible to improve the world by drawing comics

- Hey, is it now nicer over there?

© Juba