Project Management (PRO)

Kandidatekursus CS/INF

Lecture 7: Requirements

Part 2: Requirements documentation and management

Nina Boulus-Rødje (PhD), Assistant Professor
Informatics & User-Driven Innovation. Institute for People and Technologies. ninabr@ruc.dk

IEEE standard 830

- IEEE 830 provides an excellent check list, ensuring all project participants know:
  - the fundamental req. for the system to be developed
  - the data model on which development will be based
  - the system constraints

- Purpose of a req. spec. can sometimes be met through:
  - Bullet lists
  - Diagrams and Display screens
  - Characteristic examples
  - User Stories or Scenarios

  provided they include: rationale, measurability, acceptance, priority

  e.g. Do NOT make the req specification an exercise in waste of time and paper

- The usefulness of a req spec is not measured in thickness, but in <u>how well</u> the specification <u>communicates</u> to the project participants what the system should do

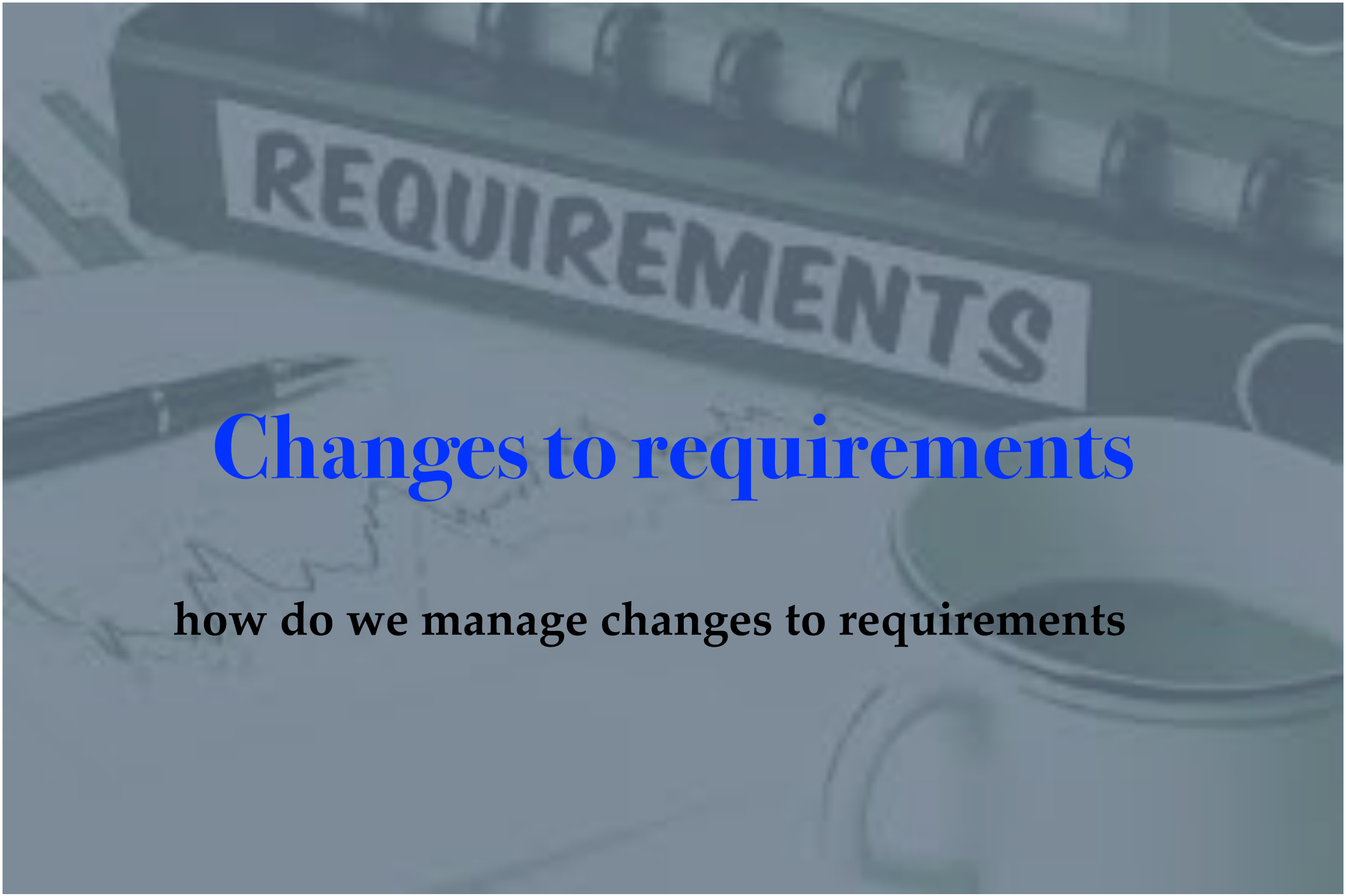
A small requirements specification document

1. Introduction
2. Purpose and goals
3. Stakeholders
   - those relevant in connection w/the req. work (e.g. not the complete list of stakeholders)
4. Context diagram
   - Interfaces of the system to the outside world
5. Requirements
   - 5.1 Functional requirements
     - Description of each functional requirement
   - 5.2 Quality requirements
     - Description of each non-functional requirement
     - Or references to the quality characteristics in the quality plan
   - 5.3 Managerial requirements
     - Description of each contextual or contractual requirement
6. Appendices

Comprehensive example of the content in a requirements list (e.g. in a spreadsheet)

- **Identification:**
  - Requirement number a unique number for each requirement for traceability
  - Version number for each requirement for traceability
- **Date for the creation or change of the requirement**
- **Initials** of the person who most recently has changed the requirement in this list
- **Status the state of the requirement** (e.g. 'Suggested', 'Ready for approval', 'Approved')
- **Source** who is the provider of the requirement (requirements stakeholder)
- **Requirement level** business-level, user-level or product-level
- **Requirement type** functional, quality or managerial
- **Description** a short and precise sentence with a controlling verb
- **Rationale** needs, expectations and reasons for the requirement
- **Measurability** quantification (amount, volume, size, accuracy, limits)
- **Acceptance criteria** how it can be determined whether the requirement is met (approved)
- **Priority** an evaluation of importance in relation to other requirements (e.g. 1-5 scale, 5 is highest)
- **Dependencies to other requirements** where this dependency is significant/critical
- **Further documentation** reference to further information about the requirement
- **Other comments** any explanatory comments and observations on the requirement
- **Change request** reference to the request for change (for change control)

[NB: The absolute minimal content in the list is marked with *]



Changes to requirements

how do we manage changes to requirements

How often do changes to requirements occur?

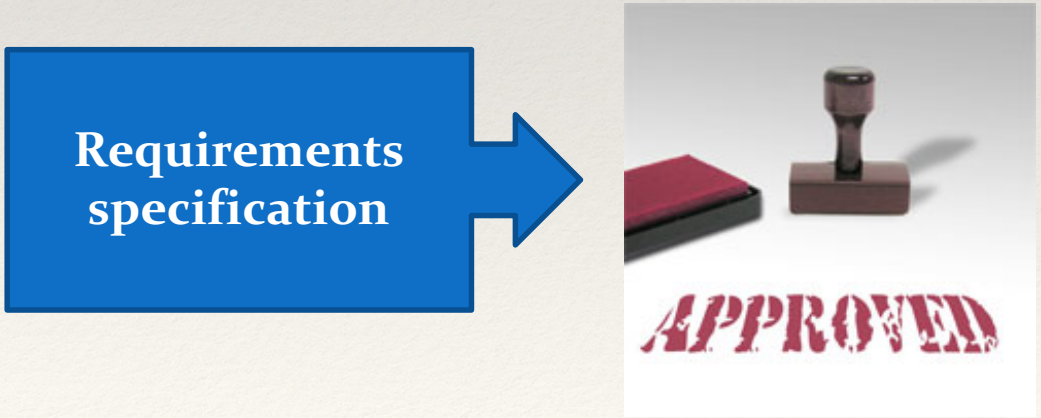
(Capers Jones, 1994)

- The average number of changes to requirements in a sample of 60 projects was 35%
  - e.g. 1/3 of the system was changed after the requirements specification had been done
- *Need a formalised way of managing changes to requirements!*

- There are, on average, 1-4% changes to requirements per month
  - e.g. 40% of the system is changed during the course of a 3-year project
  - the frequency of changes to SW req. is higher than that for e.g. the electronics or construction industry
- *The shorter the project, the fewer the changes!*

Managing changes to requirements

- Skilful project managers will often be able to increase a project's order sum via careful and consistent change management (Kousholt, 2012e: 356)
- Change management take effect as soon as the requirements specification is approved ("locked")

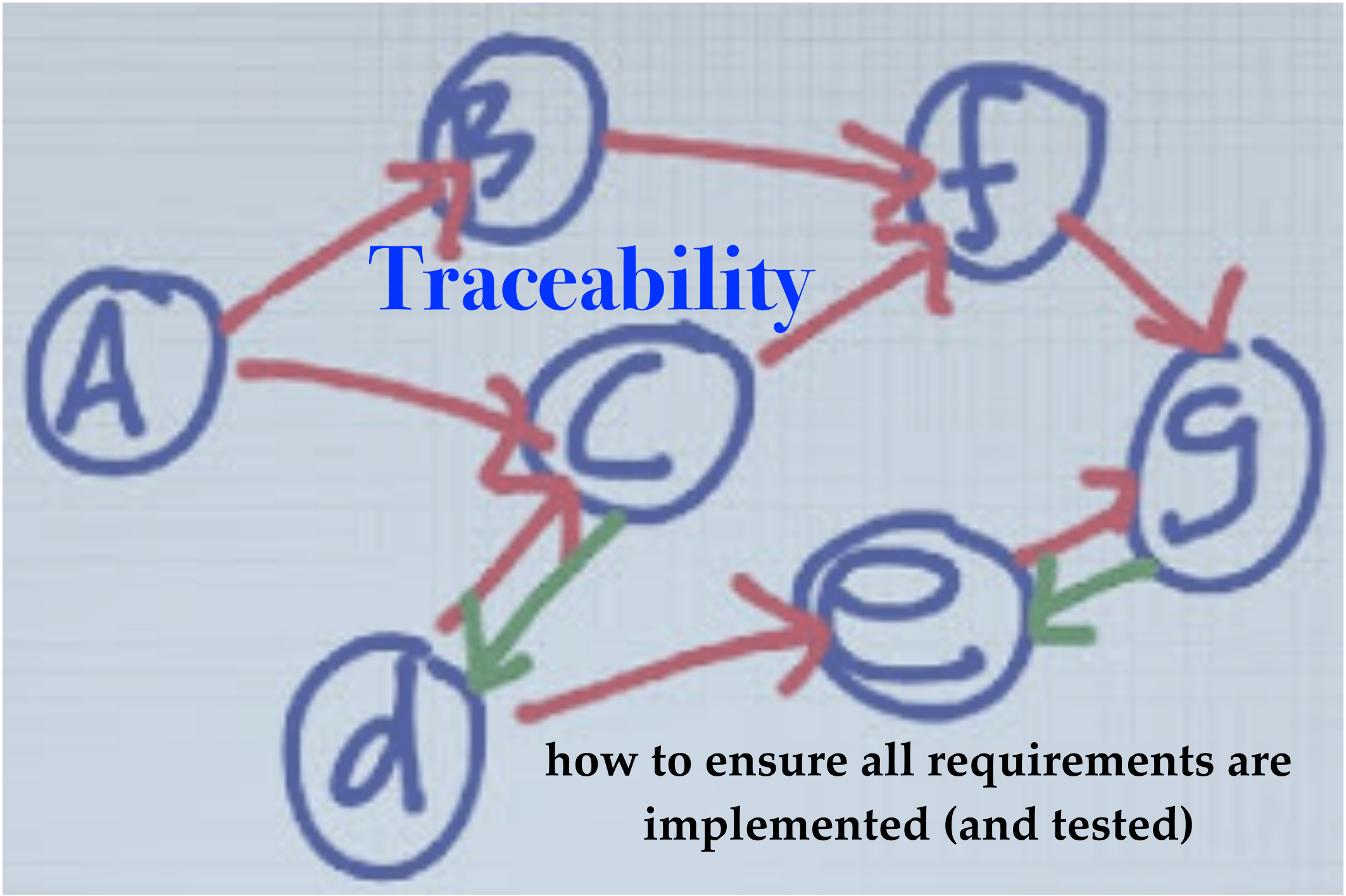


Establish a formalised process for managing requirements

Ensure that all changes to requirements are:

- subjected to the same systematic approach as the original requirements
- well documented
- handled according to a well-defined process (based on a formal request for change)
- evaluated and analysed for impacts (time/size/price/consequences)
- formally approved and the decision documented

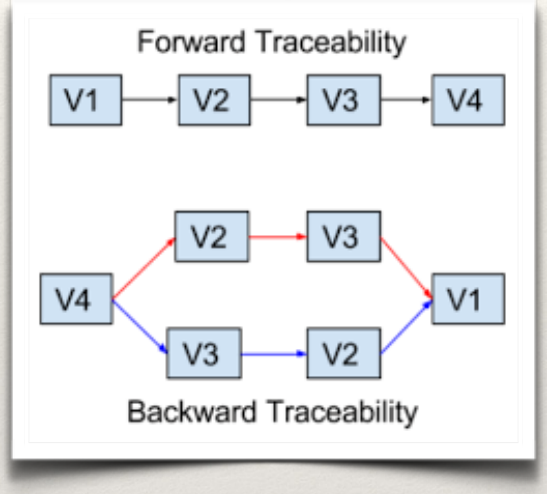
Traceability



how to ensure all requirements are implemented (and tested)

Definition of traceability

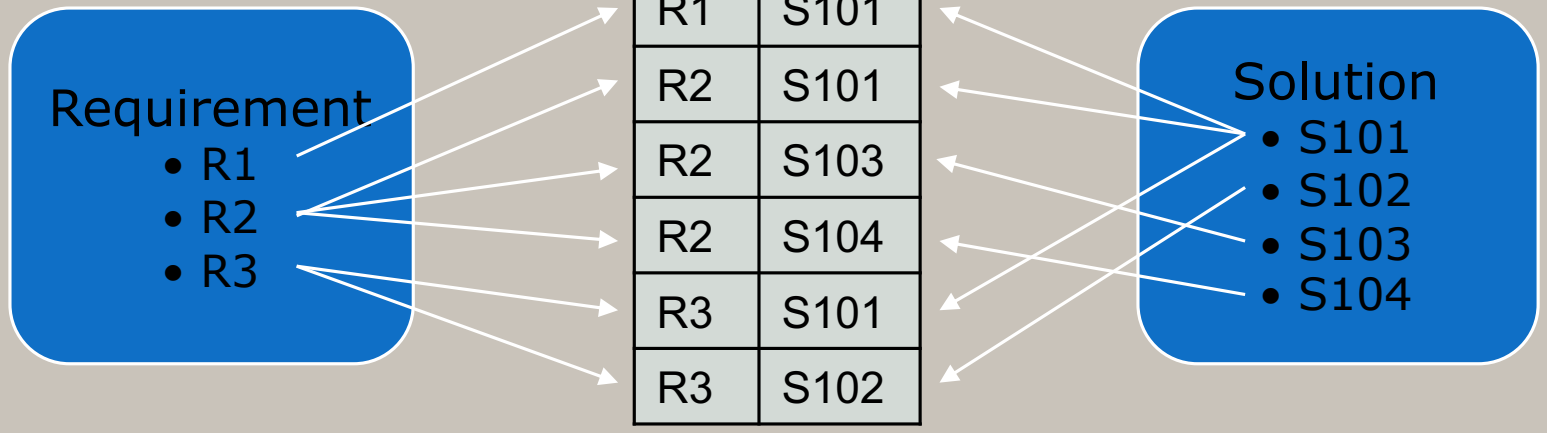
- **Traceability**: enables following the "life" of a req. (product)
  - Answer questions for e.g.:
    - where did it come from, and what did it result in
    - what did we do and why
    - what have we decided and what is implemented



- **Forwards traceability**
  - Req. can be followed all the way though to the solution (and test) with the purpose of documenting where/how the req. is realised (and covered by tests)

- **Backwards traceability**
  - Req. can be traced back from solution to their origin with the purpose of documenting where the solution came from, and that only solutions corresponding to req. are included

Traceability matrix



R1	S101
R2	S101
R2	S103
R2	S104
R3	S101
R3	S102

Forwards traceability

R1	S101
R2	S101
	S103
	S104
R3	S101
	S102

Backwards traceability

S101	R1
	R2
	R3
S102	R3
S103	R2
S104	R2

Why is traceability important?

For the project:

- Allow (distributed/large) team to keep track of what has been decided & implemented? And by whom?
- Handling conflicts during the development
- Coordinating exchange of information on requirements (and changes to them)
  - E.g. in case of outsourcing/subcontracting
- Managing evolutionary development
  - Where systems "grow" in increments

For the organization:

- Preserving organizational "memory": what did we actually do and why?
- Enabling reuse of existing solutions
- Enabling re-engineering of large old systems ("legacy systems")
- Secure process knowledge from the development of large scale systems

Examples of traceability

- **Dependency**
  - E.g. requirement no. 123 for the user interface depends on requirement no. 456 for the chosen HW platform
- **Satisfaction**
  - E.g. requirement no. 123 is met/satisfied through design no. 456
- **Enhancement**
  - E.g. requirement no. 789 was changed 2009-09-03 because of an enhancement of an earlier requirement no. 789 from 2009-01-04
- **Rationale**
  - E.g. why we decided to include requirements no. 456? What were the arguments and reasons for this choice?

Requirements elicitation

ways of finding requirements

Requirements have many faces

- Many stakeholders
  - Users, owners, developers …
- Involved in many activities
  - Identification, specification, diagramming, documentation, etc.
- Used for many things
  - Communication, contracts, development

The problem with requirements elicitation

- System requirements change over time
  - E.g. developments in technology or market needs
- It is hard to articulate requirements
  - Features and processes are difficult to describe precisely
  - End-users find it difficult to explain what they need / would like to have
- It can be difficult to motivate users
  - Maybe they don't like the idea about a new system
  - The reward – in the form of the new system – will not come until the end, maybe years from now!

Requirements- “us” and “them”

How we see the users

- Don’t know what they want
- Can’t articulate what they want
- Have too many politically motivated needs
- Can’t prioritise their needs
- Refuse to take responsibility for the system
- Are unwilling to compromise

How users see us

- Don’t understand the business
- Behave strangely to the politics in the organisation
- Try to tell us how we should do our job
- Say NO all the time
- Are always over budget
- Are always late
- Are incapable of adapting to changing needs

User defence strategies

- Kitchen sink
  - Throw everything in as requirements
- Fogging
  - Ask for more – something to negotiate about
- ‘Just the same as today, thanks’
  - Laziness or lack of knowledge?
  - Even conversion projects are never 'just' the same

Techniques for requirements elicitation

- Scenarios: Relate needs to use situations. Describe the tasks for every scenario, context/environment, task purpose and who performs the task
- Usability test of tasks using a prototype: Ensure users are able to operate the system for their regular tasks, based on a navigational prototype of the user interface
- Interviews
- Questionnaires
- Observations: workflow, how execution problems are dealt, etc.
- User discussions
- Use cases: who is the actor? Goals? Preconditions?main tasks? Exceptions? Variations? Info required? inf desired?
- Prototypes: build user interface without adding detail functionality for users to interpret the features of intended SW
- Statistics of usage
- User requests
- Workshops
- Brainstorming
- Background reading
- Survey

[see Brüel & Kjær]

Always ask **Why** 5 times

- *"If I had asked people what they wanted, they would have said faster horses"* (Henry Ford)
- Clients (stakeholders) often talk about their proposals for a solution
  - instead of their problems
- Solutions are the way they pass on their problems
  - often not realizing what their real problems are
- Therefore→ always use the technique of asking "5 times Why"
  - often 3 times are enough

Neural diagnostics

System shall have mini-keyboard with start/stop button, ...
Why?

So that it is possible to operate it with the "left hand"
Why?

Because both hands must be at the patient
Why?

To control electrodes and bandages, and to calm and reassure the patient.

[see Lauesen (2002: 29)]

Additional readings

- Carroll, J.M. (1995): Scenario-Based Design: Envisioning Work and Technology in System Development, Wiley.
- Firesmith, D. (2003): *Specifying Good Requirements*, Journal of Object Technology, vol. 2, no. 4, pp. 77-87. http://www.jot.fm/issues/issue_2003_07/column7.pdf
- Glintz, M. (2014): *A glossary of Requirements Engineering Terminology*, Version 1.6, IREB. http://www.ireb.org/fileadmin/IREB/Download/Homepage%20Downloads/IREB_CPRE_Glossary_16.pdf
- Jones, C. (1994): Assessment and Control of Software Risks, Yourdon Press.
- Kousholt, B. (2012): Projektledelse – Teori og praksis, Nyt Teknisk Forlag, København.
- Lapouchnian, A. (2005): Goal-Oriented Requirements Engineering – An Overview of Current Research, Department of Computer Science, University of Toronto.
- Lauesen, S. (2002): Software Requirements – Styles and Techniques, Addison-Wesley.
- Lauesen, S.(*2007*): Guide to Requirements SL-07 - Template with Examples. Lauesen Publishing http://www.it-c.dk/people/slauesen/Papers/RequirementsSL-07v5.4.docx
- Lauesen, S. & O. Vinter (2001): *Preventing Requirement Defects: An Experiment in Process Improvement*, Requirements Engineering Journal, Vol. 6 No. 1, Springer-Verlag London. (www.ottovinter.dk/PrevDefectsREJ.pdf)
- Mikkelsen, H. & J. O. Riis (2013): Project Management - multiperspective leadership, Prodevo, ISBN-13: 9788789477220
- Molich, R. (1996): *Brugervenlige edb-systemer*, Teknisk Forlag, København.
- Scharer L. (1981): *Pinpointing Requirements*, Datamation, Vol. 27, No. 4.
- Vinter O. & S. Lauesen (2000): *Analyzing Requirements Bugs*, Bug Report Department in Software Testing & Quality Engineering Magazine, Vol. 2-6, Nov/Dec 2000. (www.ottovinter.dk/stqema26.doc)
- Vinter O., S. Lauesen & J. Pries-Heje (1999): *A Methodology for Preventing Requirements Issues from Becoming Defects*, ESSI Project 21167, Final Report, Brüel & Kjær Sound & Vibration Measurement A/S, Nærum. (www.ottovinter.dk/Finalrp3.doc)