cc BY NC SA

Software Engineering

Lecture 12 – Requirements Engineering

© 2015-19 Dr. Florian Echtler
Bauhaus-Universität Weimar
<florian.echtler@uni-weimar.de>

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

BY NC SA

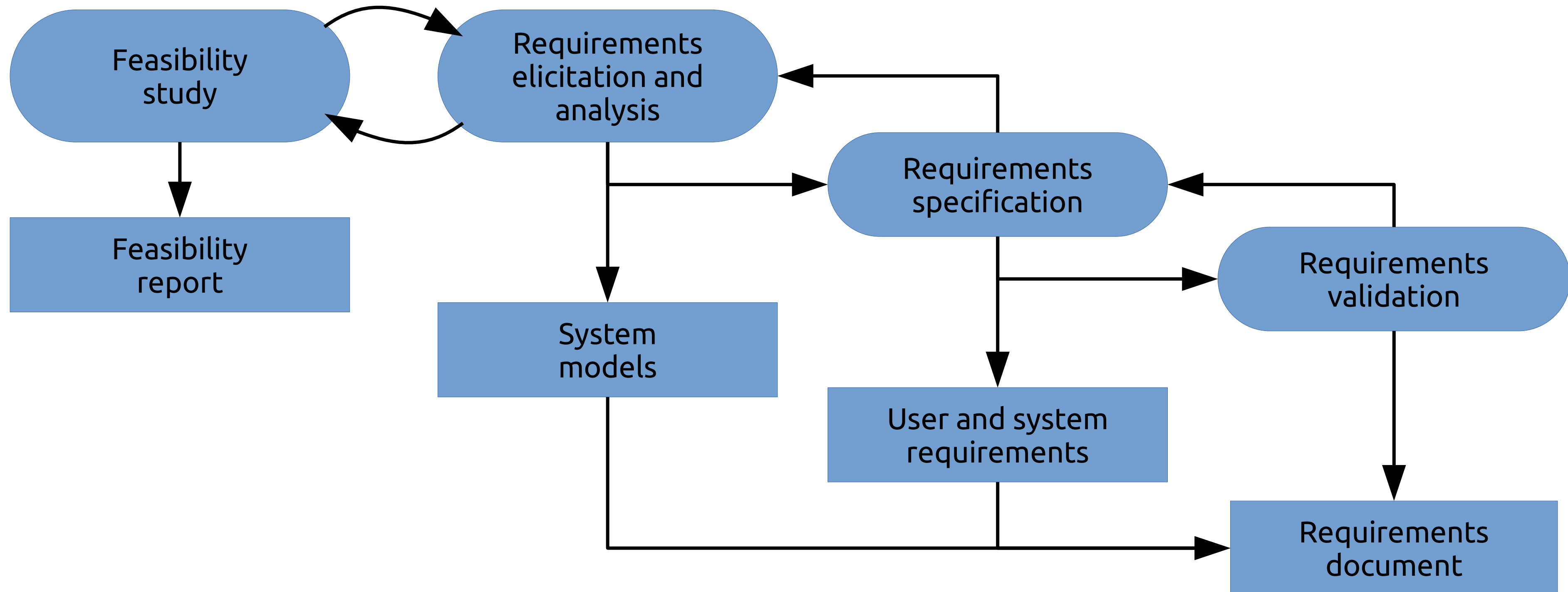
Requirements Engineering

- Also called *software specification*
- Define functionality of/constrains on the software product
- Sub-activities:
  - Feasibility study
  - Requirements elicitation/analysis
  - Requirements specification
  - Requirements validation

CC BY NC SA

Software Specification Process

Image source (FU): Sommerville, Software Engineering, Chapter 2



BY NC SA

Feasibility study

- Short, focused evaluation
- 3 major questions:
  - Does the system contribute to the overall objectives of the organization?
  - Can the system be implemented within schedule and budget (*) using current technology?
  - Can the system be integrated with other systems that are used?

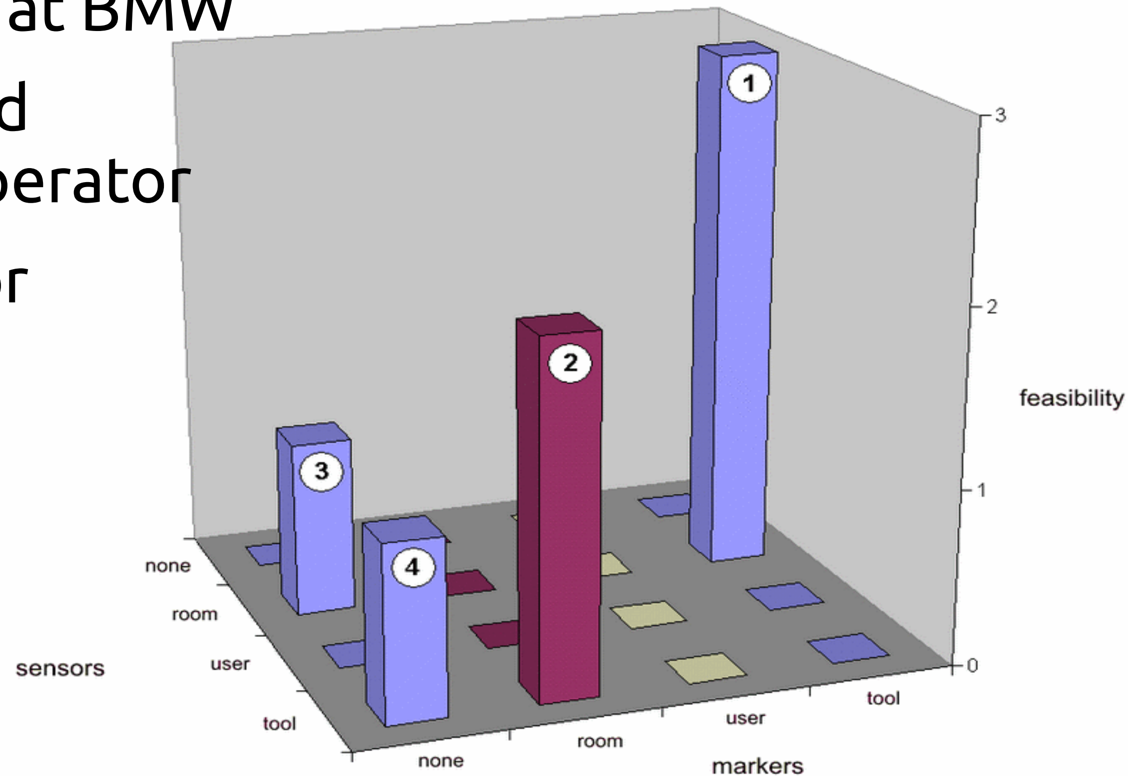
(*) definite answer probably impossible

CC BY NC SA

Feasibility study: example

Image source (FU): Echtler et al., “The Intelligent Welding Gun”, 2003

- Augmented reality project at BMW
- Goal: track welding gun and display bolt positions to operator
- Analysis of design space for sensor/marker placement



CC BY NC SA

Requirements elicitation

- User doesn't really know …
  - what they actually need
  - what is possible
- Developer doesn't really know …
  - the problem
  - the context

CC BY NC SA

Requirements elicitation (2)

- Important first steps for developers:
  - Identify stakeholders (users, managers, admins, …)
  - Learn about fundamental context properties
- Iterative process:
  - Discovery (e.g. by interviews, ethnography, …)
  - Classification & organization (e.g. through model architecture)
  - Prioritization & negotiation (conflict resolution between stakeholders)
  - Specification & documentation

CC BY NC SA

Example: stakeholders for an ATM?

- Bank customers
- Bank managers
- Counter staff
- Database administrators
- Security managers
- Representatives of other banks
- Marketing department
- Hardware and software maintenance engineers
- Banking regulators

CC BY NC SA

Requirements discovery

- Interviews with stakeholders
  - Closed/Open (with/without predefined questions)
  - Hybrid (semi-structured) → most common
  - Not suitable for domain requirements (too familiar for interviewee, not familiar for developer)
- Ethnography
  - Observational technique, immersion in the work environment/context in which the product is used
  - Helps to discover implicit requirements, constraints and social/environmental details

CC BY NC SA

Requirements classification

- E.g. using *viewpoints*
- Represent perspectives of different stakeholders
  - Interactor viewpoint – persons/systems which interact directly with the product, e.g. users
  - Indirect viewpoint – no direct interaction but influence on requirements, e.g. managers
  - Domain viewpoint – internal regulations, legal requirements etc.

cc BY NC SA

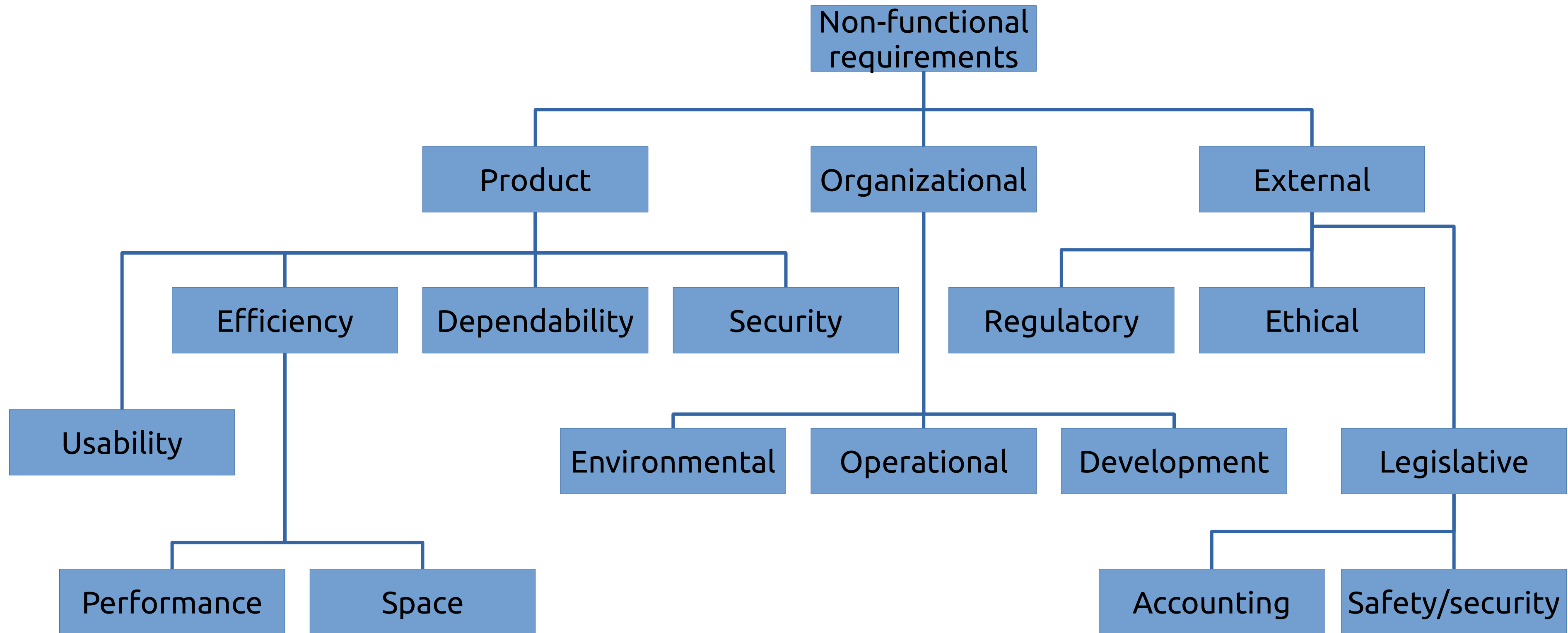
Requirements classification (2)

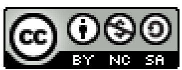
- Functional requirements
  - What should the system (not) do?
- Non-functional requirements
  - Reliability, response time, security, ease of use …
  - May lead to additional functional requirements
- Domain requirements (variant of NFR)
  - Regulations, laws etc.
  - May lead to additional (non-)functional requirements

CC BY NC SA

Requirements classification (3)

Image source (FU): Sommerville, Software Engineering, Chapter 4





Requirements description

Source (FU): Sommerville, Software Engineering, Chapter 4

- “It is natural for a system developer to interpret an ambiguous requirement in a way that simplifies its implementation. Often, however, this is not what the customer wants.”

  → path of least resistance

  → requirements must be as precise as possible

CC BY NC SA

Types of requirements description

- Sentences in natural language (+ diagrams)
  - user/system requirements document (“Lasten-/Pflichtenheft”)
- Structured language – natural language using structured template
- Design description language/graphical notations, e.g. UML & related notations
- Mathematical specifications

CC BY NC SA

Natural language RD

Source (FU): Sommerville, Software Engineering, Chapter 4

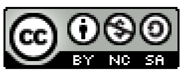
- Example: insulin pump
- Format: 1 sentence requirement + rationale
- Mandatory → “shall”, optional → “should”
- “The system shall measure the blood sugar and deliver insulin, if required, every 10 minutes. (Changes in blood sugar are relatively slow so more frequent measurement is unnecessary; less frequent measurement could lead to unnecessarily high sugar levels.)”
- Still room for interpretation …

CC BY NC SA

Structured language RD

Source (FU): Sommerville, Software Engineering, Chapter 4

Attribute	*Description*
Function	Compute insulin dose: safe sugar level.
Description	Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.
Inputs	Current sugar reading (r2), the previous two readings (r0 and r1).
Source	Current sugar reading from sensor, other readings from memory.
Outputs	CompDose – the dose in insulin to be delivered.
Destination	Main control loop.
Action	CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result is rounded to zero then CompDose is set to the minimum dose that can be delivered.
Requirements	Two previous readings so that the rate of change of sugar level can be computed.
Pre-condition	The insulin reservoir contains at least the maximum allowed single dose of insulin.
Post-condition	r0 is replaced by r1, then r1 is replaced by r2
Side effects	None.



Mathematical RD

Source (FU): Sommerville, Software Engineering, Chapter 4

Condition	*Action*
Sugar level falling (r2 < r1)	CompDose := 0
Sugar level stable (r2 = r1)	CompDose := 0
Sugar level increasing (r2 > r1) and rate of increase decreasing ((r2 – r1) < (r1 - r0))	CompDose := 0
Sugar level increasing (r2 > r1) and rate of increase stable or increasing ((r2 – r1) >= (r1 - r0))	CompDose := round((r2 – r1) / 4) if CompDose = 0: CompDose := MinimumDose

cc BY NC SA

Requirements Validation

- Check document for …
  - Validity/Completeness – are all requirements correctly met?
  - Consistency – are there conflicts between requirements?
  - Realism – can requirements be implemented with given resources?
  - Verifiability – can requirements be checked for completion?

cc BY NC SA

Requirements Validation (2)

- Check through …
  - Reviews – systematic manual analysis, preferably by external reviewers
  - Prototyping – create prototypes conforming to current requirements, review with stakeholders
  - Test-case generation – create test cases or test procedures for requirements

cc BY NC SA

Requirements Validation (3)

- Testing non-functional requirements:
  - Speed: transactions/second, response time, …
  - Ease of use: training time, size of manual, …
  - Reliability: uptime, mean time between failures, …
  - Robustness: time to restart after failure, probability of data corruption, …

BY NC SA

Relation to software processes

- In "traditional" processes
  - RE mostly done at the start
  - multiple cycles, but finishes with req. document
- In agile processes
  - interleaved with development
  - feasibility study probably still initial step
- Hybrid approach: e.g. RE interleaved with prototyping

CC BY NC SA

Questions/Comments?