Requirements Engineering I

Chapter 2

Fundamentals

Chapter roadmap

2.1 Nine Principles

Foundational facts and insights

2.2 Classification

In which ways can we classify requirements and what does this help

2.3 Shared Understanding

Why we need it and how we achieve it

2.4 Context

Why context matters in RE and how to consider it

2.5 Requirements and Design

How do they relate to each other?

2.1 Principles of Requirements Engineering

Nine basic principles

1 Value-orientation

2 Stakeholders

3 Shared understanding

4 Context

5 Problem – Requirement – Solution

6 Validation

7 Evolution

8 Innovation

9 Systematic and disciplined work

Principle 1: Value-orientation

Requirements are a means, not an end.

Traditional Requirements Engineering: always write a complete specification

However...

- Customers typically pay for systems, not for requirements
- Many successful projects don't have a complete specification
- Good Requirements Engineering must create value
- Value comes indirectly

Requirements are a means, not an end

[Glinz 2008]

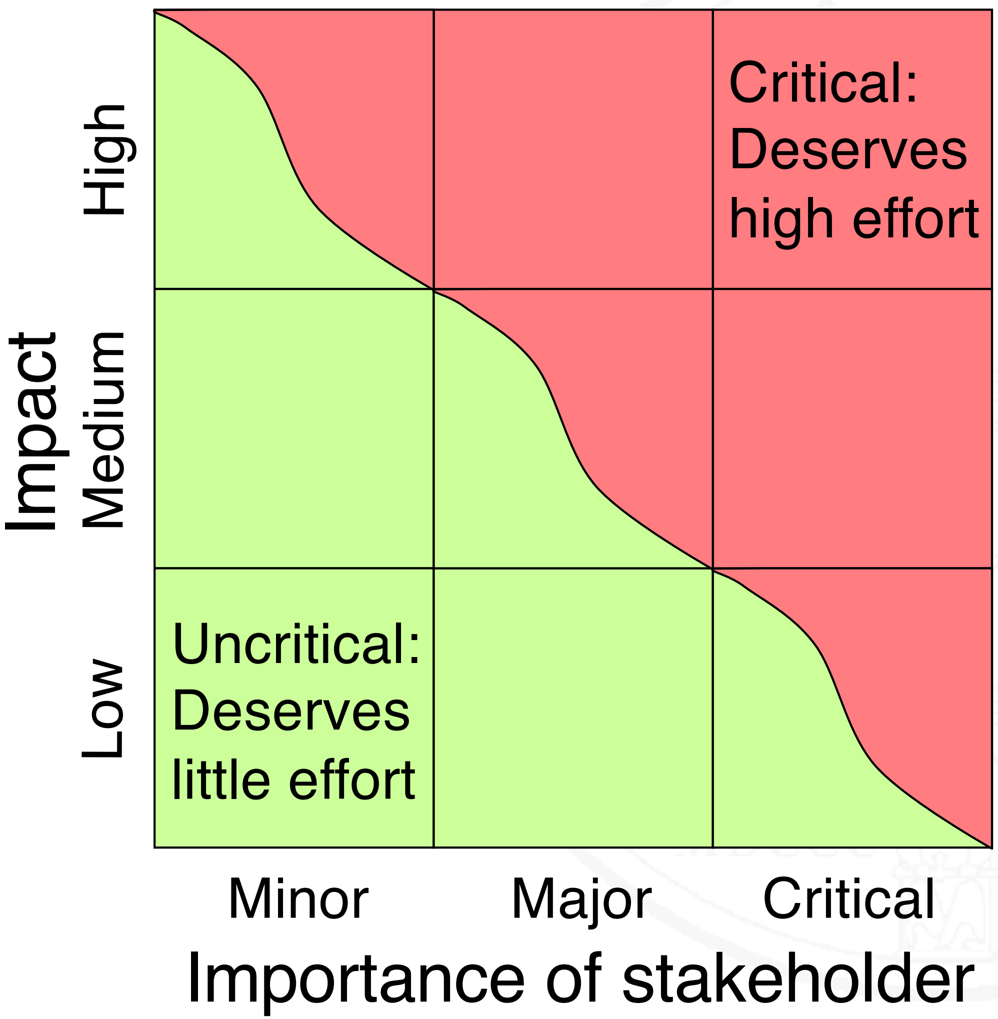
- Requirements shall deliver value
- Value of a requirement:
  - The benefit of reducing development risk
    (i.e. the risk of not meeting the stakeholders' desires and needs)
  - minus the cost of specifying the requirement

☞ Adapt the effort put into RE such that the specification yields optimum value

- Low risk: little RE　　　High risk: full-fledged RE

☞ Assessment of value requires assessment of risk

Assessing risk



[Glinz 2008]

- Assess the criticality of the requirement
- Consider other factors (next slide)
- Use requirements triage techniques

Assessing risk: other factors

- Specification effort
- Distinctiveness
- Shared understanding
- Reference systems
- Length of feedback-cycle
- Kind of customer-supplier relationship
- Certification required

The effort invested into requirements engineering shall be inversely proportional to the risk that one is willing to take.

Principle 2: Stakeholders

RE is about satisfying the stakeholders' desires and needs.

Who is "the customer"?

In our sample problem: The resort owner? The skiers?

In reality: Many persons in many roles are involved

DEFINITION. Stakeholder – A person or organization who influences a system's requirements or who is impacted by that system.

[Glinz and Wieringa 2007]
[Glinz 2024]
[Macaulay 1993]

Note that influence can also be indirect.

Viewpoints

The same building.
Different views.

Different viewpoints by different stakeholders must be taken into account.

[Nuseibeh, Kramer und Finkelstein 2003]

Consensus and variability

The viewpoints and needs of different stakeholders may conflict

Requirements Engineering implies

- Discovering conflicts and inconsistencies
- Negotiating
- Moderating
- Consensus finding

But: also determine where variability is needed

For stakeholder identification and management, see Chapter 4

Principle 3: Shared understanding

Successful systems development is impossible without a common basis.

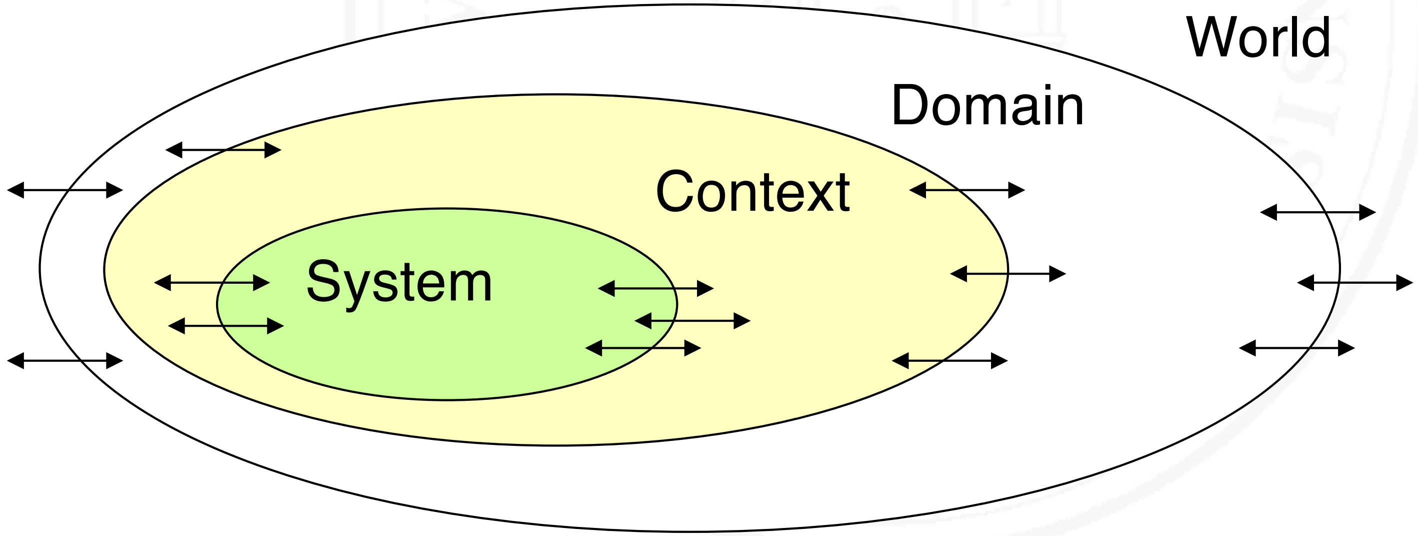
- A basic prerequisite for any successful development of systems
- Created, fostered and assured in Requirements Engineering

→ Chapter 2.3

Principle 4: Context

Systems cannot be understood in isolation.

Systems are always embedded in a context.

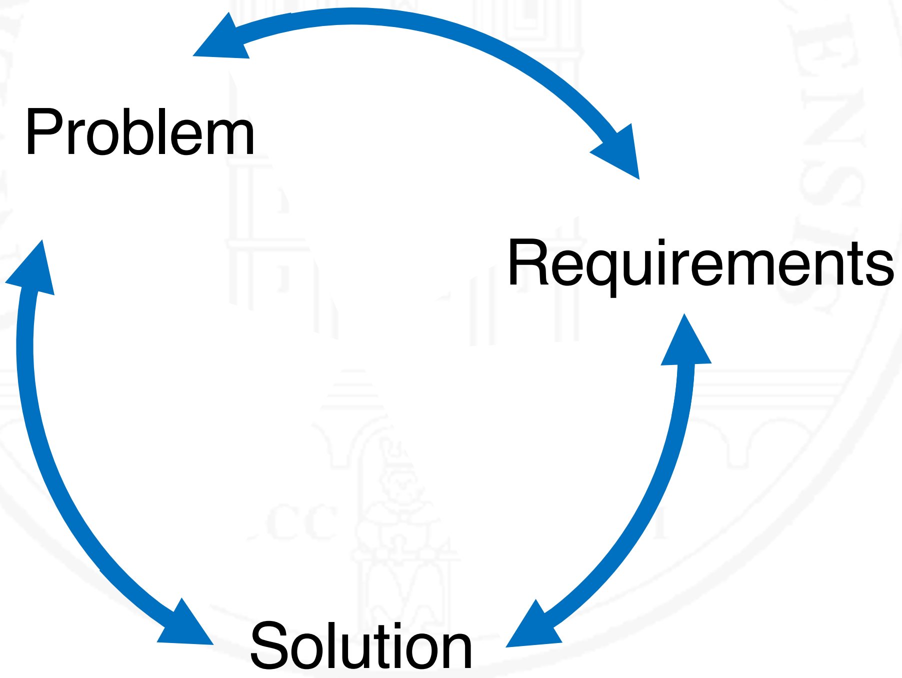


→ Chapter 2.4

Principle 5: Problem – requirement – solution

An inevitably intertwined triple.

Having a problem, we need requirements for a system that solves the problem.



Specification-Implementation Intertwinement

[Swartout and Balzer 1982]

- Traditional Requirements Engineering: the waterfall
  - Start with a complete specification of requirements
  - Then proceed to designing and implementing a solution
- Does not work properly in most cases
- Specification and implementation are inevitably intertwined:
  - Hierarchical intertwinement: high-level design decisions inform lower-level requirements
  - Technical feasibility: non-feasible requirements are useless
  - Validation: what you see is what you require

Requirements vs. solution decisions

The system shall provide effective access control to the resort's chairlifts.

A requirement

Manual control | Automatic control

Potential solution decisions

Requirements about selecting and training people

Requirements about turnstiles, access cards, and control software

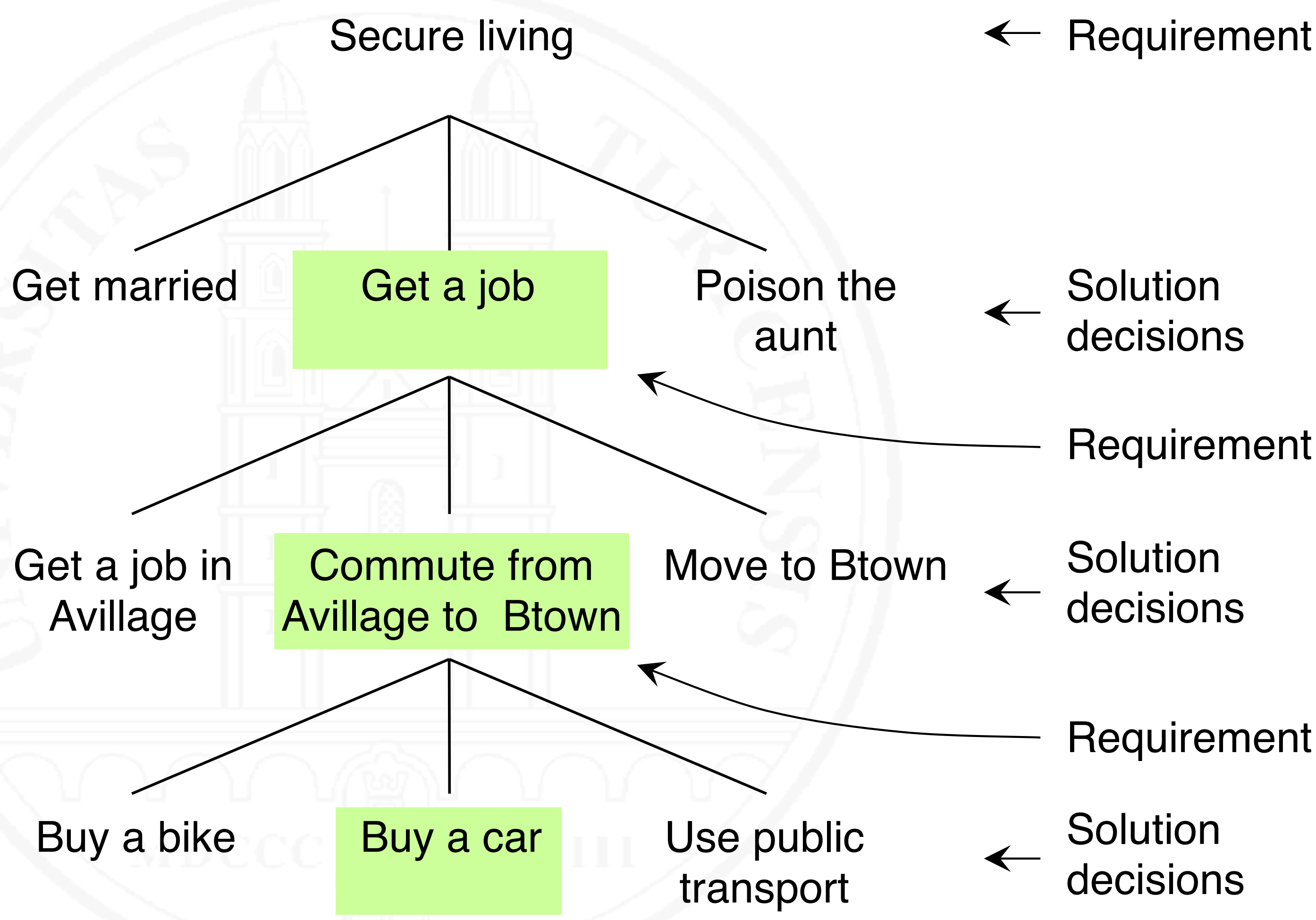
Lower level requirements

⇨ Solution decisions inform lower level requirements

⇨ Requirements and solutions are inevitably intertwined

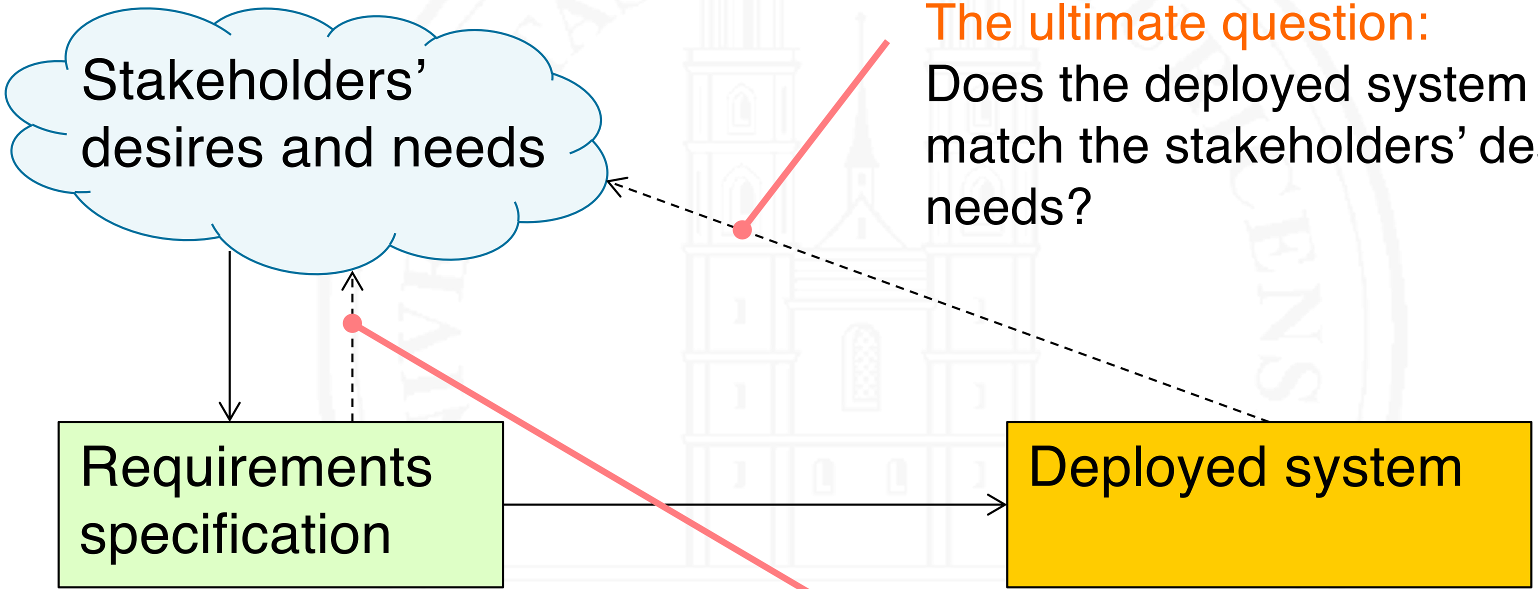
Requirements vs. solution decisions

Problem: Sonja Müller has completed her university studies and does no longer receive any money from her parents. Hence, she is confronted with the requirement to secure her living. She is currently living in Avillage and has a job offer by a company in Btown. Also, she has a rich boy friend and she is the only relative of an equally rich aunt.



Principle 6: Validation

Non-validated requirements are useless.



The ultimate question:
Does the deployed system actually match the stakeholders' desires and needs?

The risk-reduction question:
Do the documented requirements match the stakeholders' desires and needs?

Every requirement needs to be validated

→ Chapter 9

Principle 7: Evolution

Changing requirements are no accident, but the normal case.

The world evolves.

So do requirements.

The problem:

Keeping requirements stable...

... while permitting requirements to change

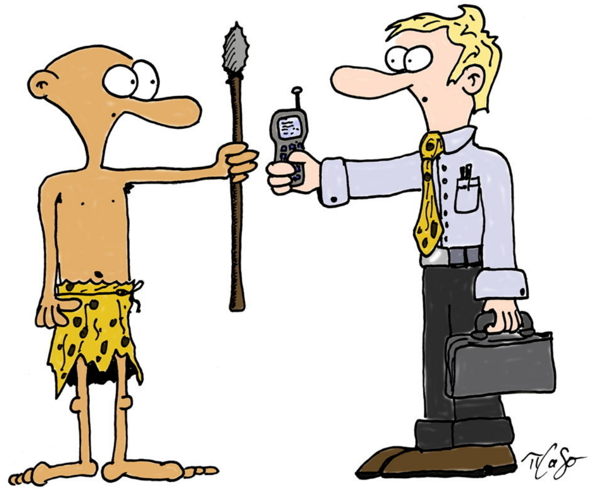


Image © C. Sommer /EKHN

Potential solutions

- Very short development cycles (1-6 weeks)
- Explicit requirements change management

→ Chapter 10

Principle 8: Innovation

Image © Apple

More of the same is not enough.

"Give the customers exactly what they want."

Maybe the worst you can do onto them.

"We know perfectly well what is good for the customer."

Your customers will love you for your attitude.

"Our new system does all the rubbish we did manually before.
But it's much faster now."

Wow, what a progress.

- Don't just automate.
- Satisfying stakeholders is not enough: strive for making them happy and excited

→ Chapter 4

Principle 9: Systematic and disciplined work

We can't do without in RE.

Requirements need to be elicited, documented, validated and managed systematically

- using a suitable process
- with suitable practices

Also applies for agile development, just with a different process and maybe different practices

Systematics does not mean "One size fits all"

- Adapt your processes and practices to the problem
- No unreflected reuse of RE techniques from previous projects

2.2 Classifying requirements

The turnstile control software shall count the number of ‘unlock for a single turn’ commands that it issues to the controlled turnstile.

A function

The operator shall be able to run the system in three modes: normal (turnstile unlocked for one turn when a valid card is sensed), locked (all turnstiles locked), and open (all turnstiles unlocked).

A behavior

The system shall be deployed at most five months after signing the contract.

A project requirement

The system must comply with the privacy law of the country where the resort is located.

A legal constraint

The reaction time from sensing a valid card to issuing an 'unlock for a single turn' command must be shorter than 0.5 s.

A performance attribute

The system shall be highly available.

A quality attribute

Requirements have a concern

Application order ↓

Question	Kind of requirement
Was this requirement stated because we need to specify ...	
... some of the system's behavior, data, input, or reaction to input stimuli – regardless of the way this is done?	functional requirement
... restrictions about timing, processing or reaction speed, data volume or throughput?	performance requirement
... a specific quality that the system or a component shall have?	specific quality requirement
... any other restriction about what the system shall do, how it shall do it, or any prescribed solution or solution element?	constraint

Kinds of requirements

DEFINITION. Functional requirement – A requirement concerning a result or behavior that shall be provided by a function of a system.

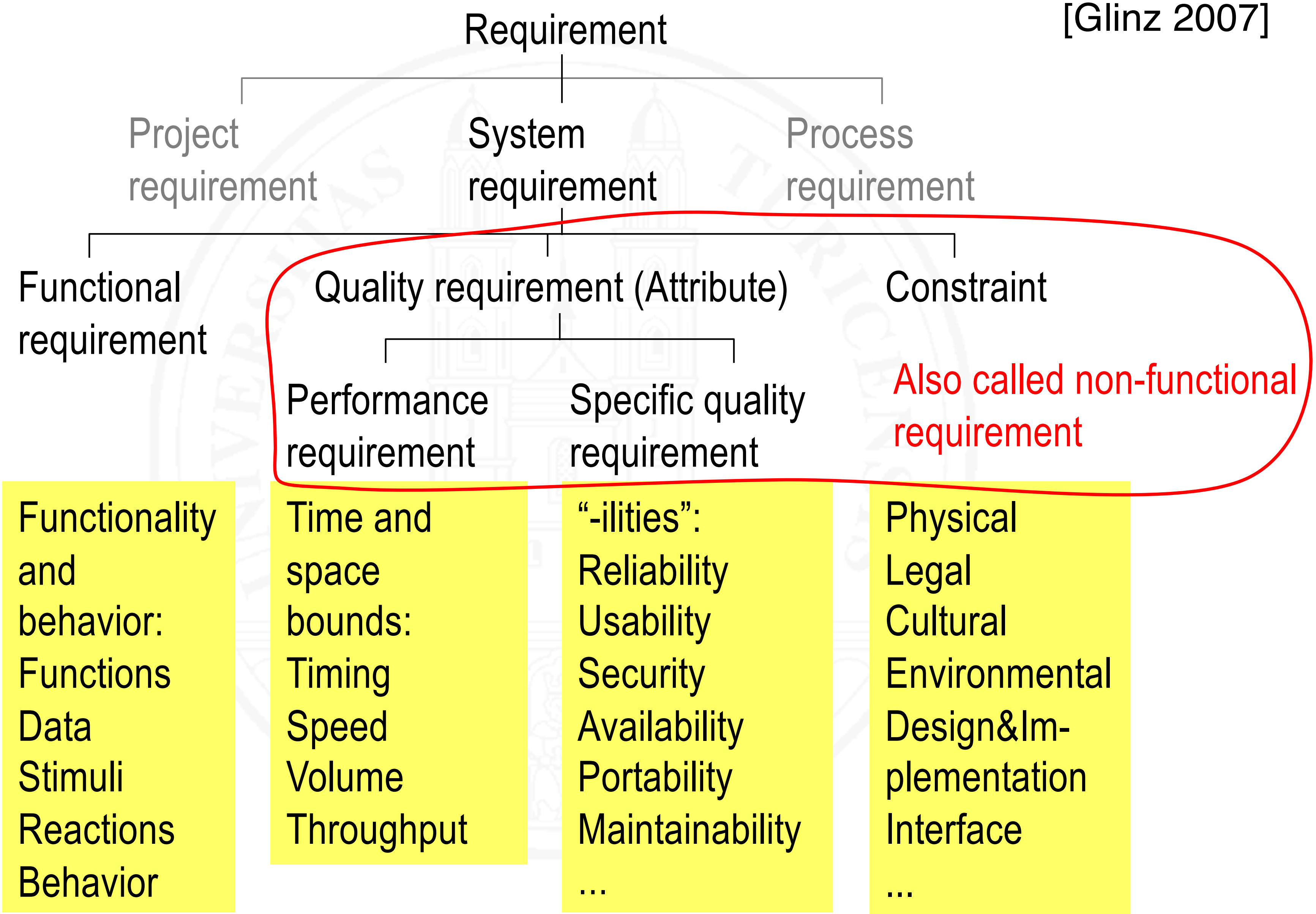
DEFINITION. Quality requirement – A requirements that pertains to a quality concern that is not covered by functional requirements

Can be sub-classified into:
- performance requirement
- specific quality requirement

DEFINITION. Constraint – A requirement that limits the solution space beyond what is necessary for meeting the given functional requirements and quality requirements.

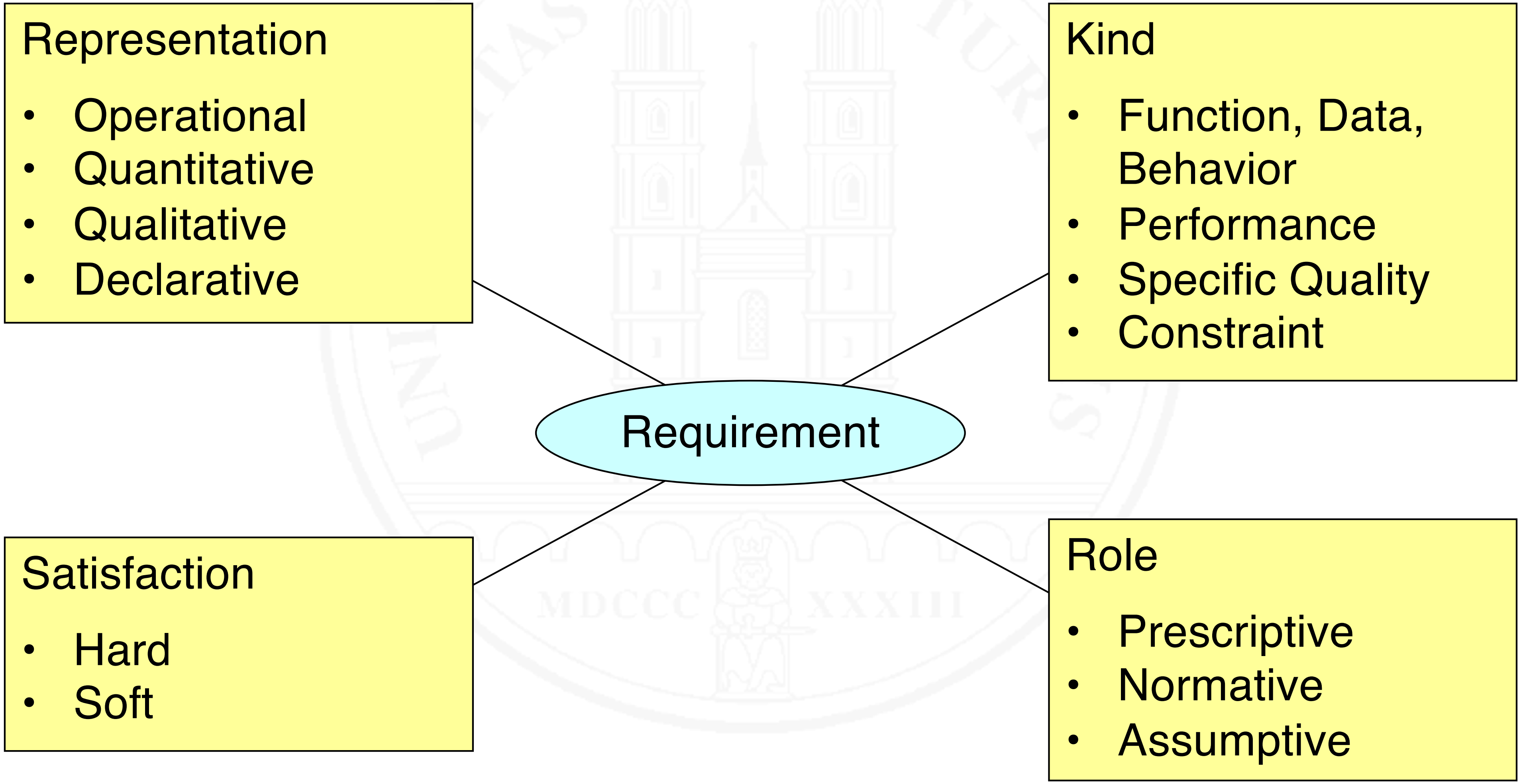
Classification according to kind

[Glinz 2007]



Beyond kind: A faceted classification

[Glinz 2005b, 2007]



Classification according to representation

The system shall be highly available. **Qualitative**

During the operating hours of the chair lift, the system must be available for 99.99% of the time. **Quantitative**

The system must comply with the privacy law of the country where the resort is located. **Declarative**

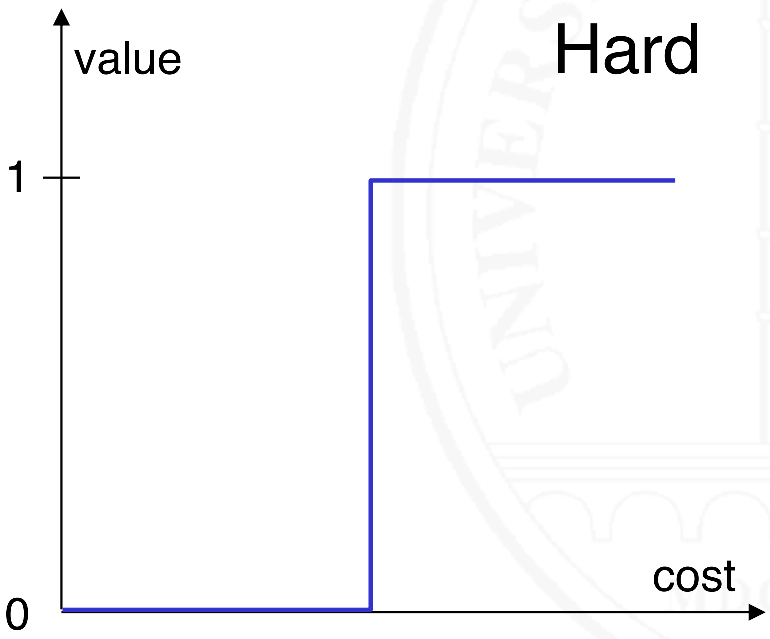
The turnstile control software shall count the number of ‘unlock for a single turn’ commands that it issues to the controlled turnstile. **Operational**

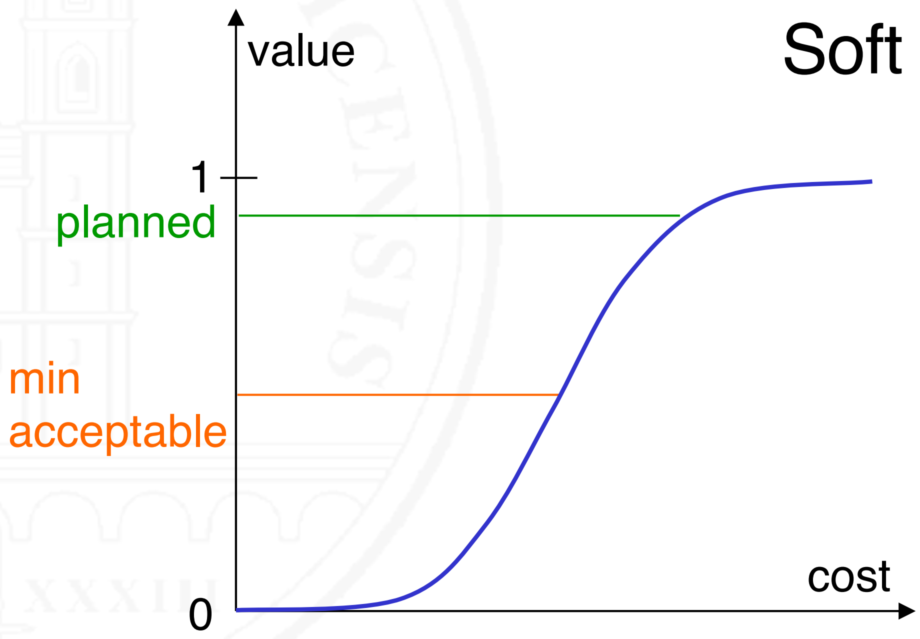
Representation informs validation

Representation	Validation technique(s)
Operational	Test, Review, Formal verification
Quantitative	Measurement
Qualitative	No direct validation technique. Use • Stakeholder judgment • Prototypes • Indirect validation by derived metrics
Declarative (informally)	Review
Declarative (formally)	Review, Model checking

Classification according to satisfaction

- Hard – The requirement is satisfied totally or not at all
- Soft – There is a range of satisfaction





Binary acceptance criterion

Range of acceptable values

Classification according to role

Prescriptive: "Classic" requirement pertaining the system-to-be

"The sensor value shall be read every 100 ms."

Normative: A norm in the system environment that is relevant for the system-to-be

"The social security number uniquely identifies a patient."

Assumptive: Required behavior of an actor that interacts with the system-to-be

"The operator shall acknowledge every alarm message."

→ Makes norms and assumptions explicit

Why do we classify?

- By kind: better understand the nature of a requirement
- By representation: how to validate a requirement
- By satisfaction: knowing what to strive for
- By role: separating context and system requirements

How to classify when in doubt

In every facet, classification is intended to be disjunctive.

However,

- A single requirement may comprise functional and quality aspects:

  When a power leak is sensed, the system shall disconnect power within 30 ms.

- For operationally represented requirements, classification can be tricky:

  The system shall protect user profiles with a two-phase login.

When in doubt

- Principled: Classify according to the dominant concern
- Pragmatic: Multi-classify if your tool permits it

Mini-Exercise

Classify the following requirements with respect to their kind, representation, satisfaction and role.

(a) When the system is in normal mode, the system shall unlock a turnstile for a single turn if the turnstile's sensor unit senses a valid access card.

(b) The system shall be compliant with GDPR.

2.3 Shared understanding

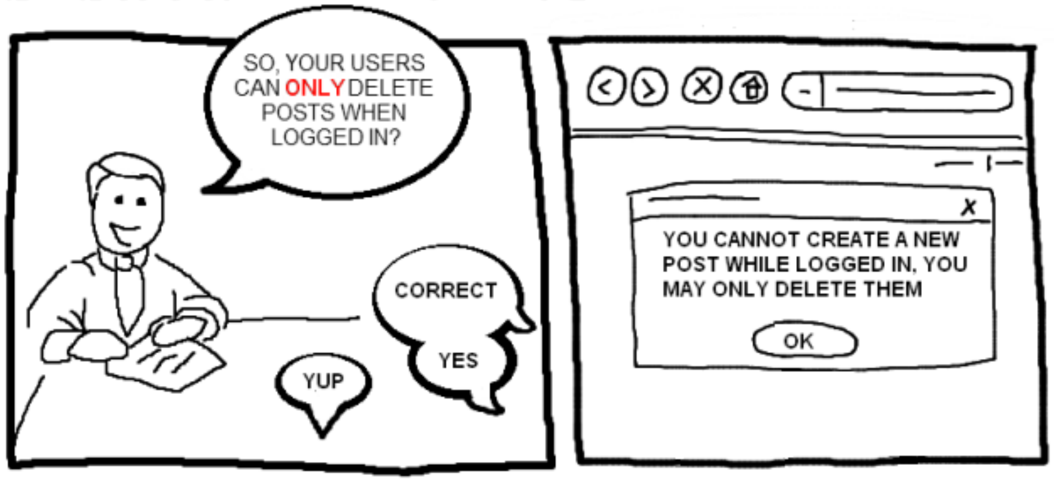
[Glinz and Fricker 2015]

Two disturbing observations:

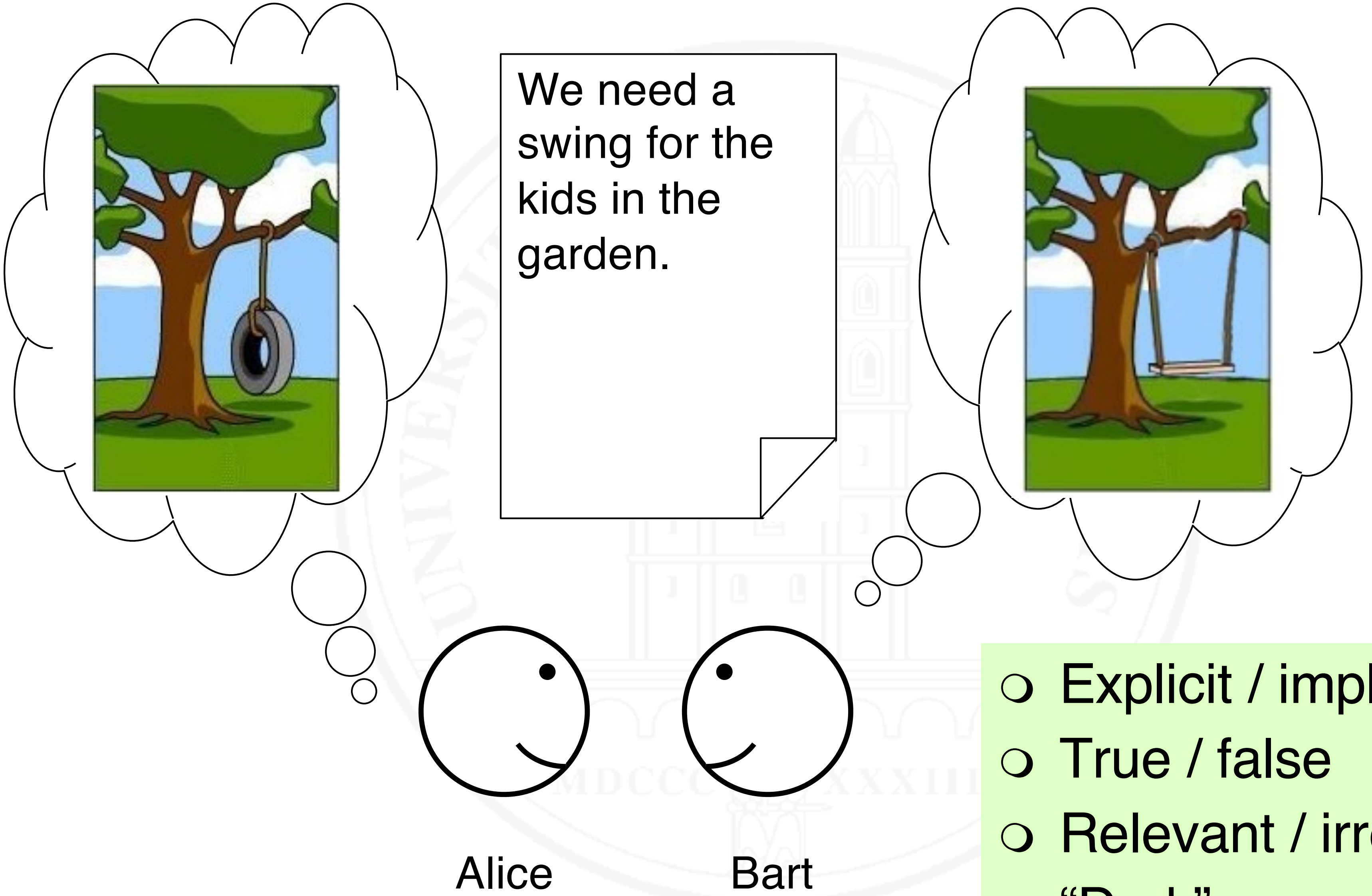
- Specifying everything explicitly is impossible and infeasible
- Explicitly specified requirements may be misunderstood

→Requirements Engineering has to deal with the problem of shared understanding

- How do we establish shared understanding?
- How can we rely on shared understanding?

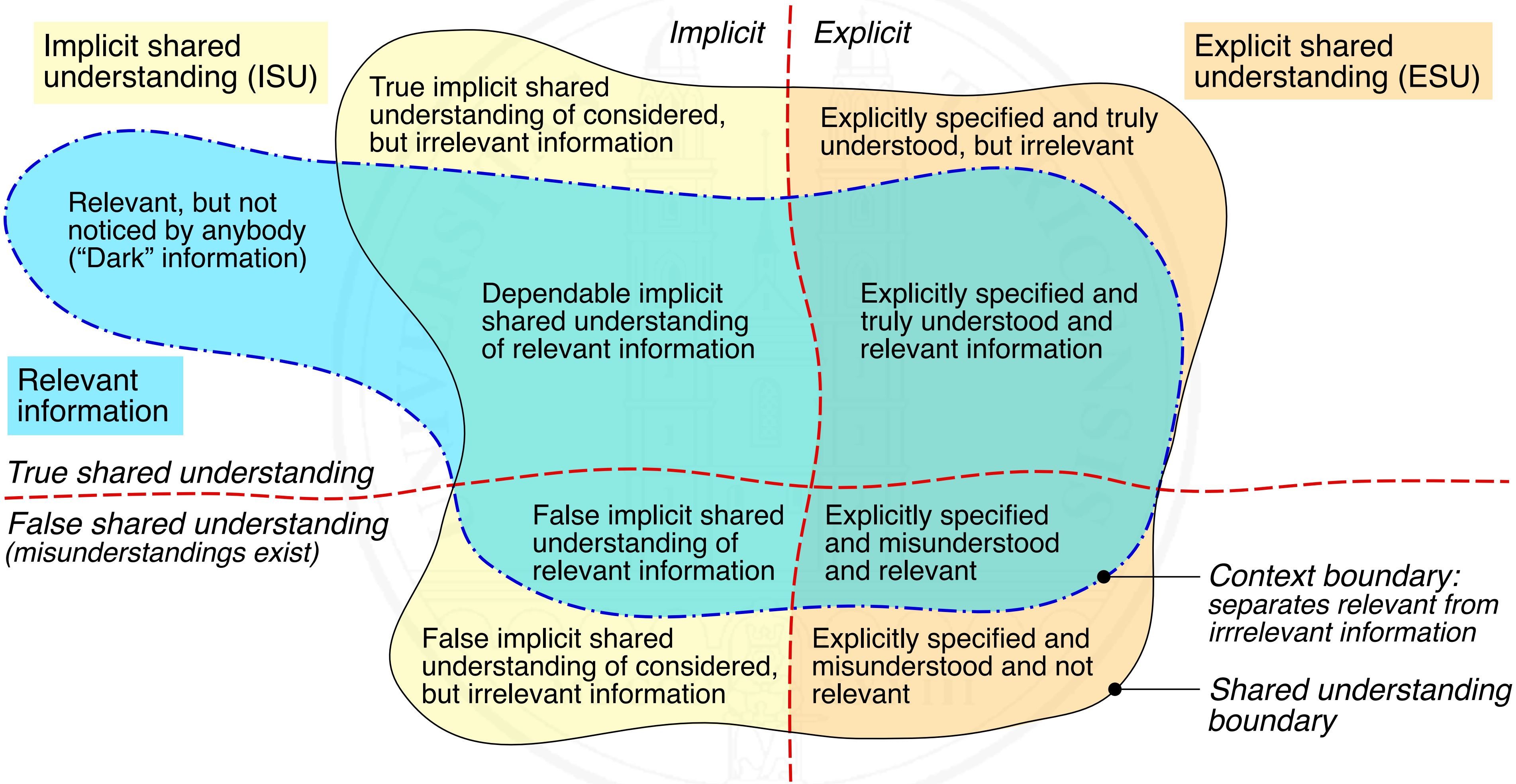


Shared understanding: the problem



- Explicit / implicit
- True / false
- Relevant / irrelevant
- “Dark”

Forms of shared understanding



[Glinz and Fricker 2015]

Rephrasing the problem

Achieve successful software development by:

(P1) Achieving shared understanding by explicit specifications as far as needed,

(P2) Relying on implicit shared understanding of relevant information as far as possible,

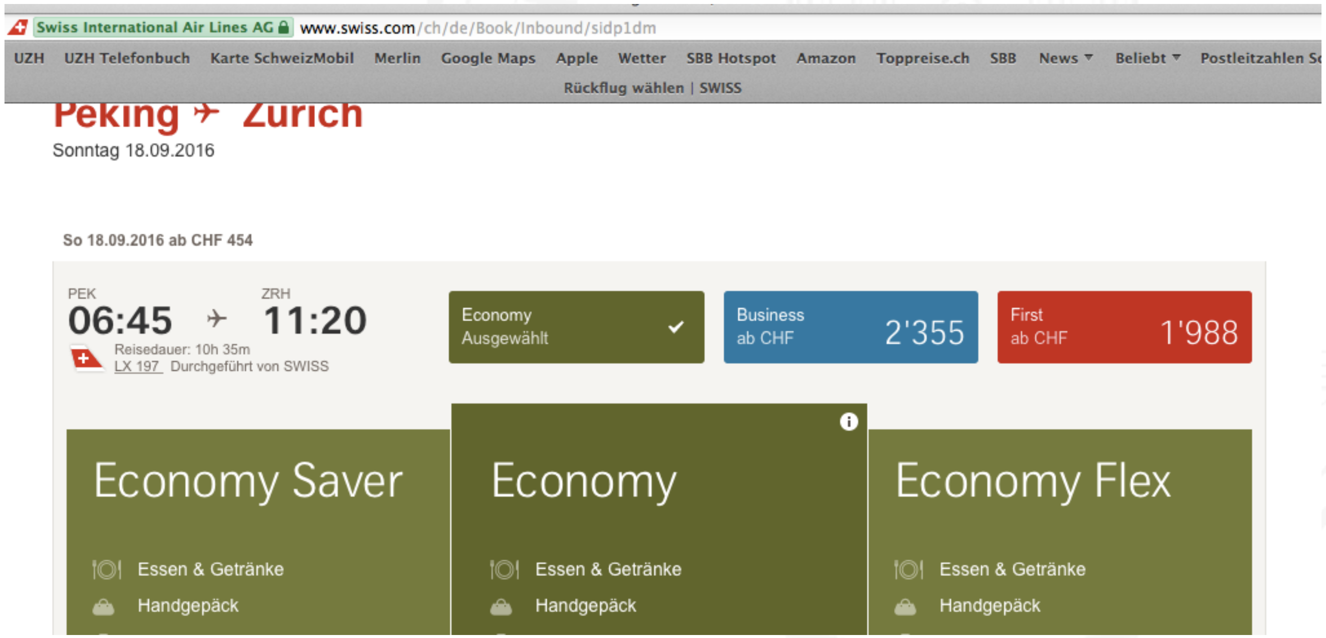
(P3) Determining the optimal amount of explicit specifications, i.e., striking a proper balance between the cost and benefit of explicit specifications.

Note that P1, P2 and P3 are not orthogonal

Shall we specify obvious requirements?

An obvious requirement for an airline ticketing system:

For every route, the following price rule shall hold:
Eco Saver < Eco < Eco Flex < Biz Saver < Biz < First



It depends on the value of the requirement.

In fact a value problem (cf. Principle 1 in this chapter)

How can we achieve specifications that create optimal value?

Value means

- The benefit of an explicit specification

  Bringing down the probability for developing a system that doesn’t satisfy its stakeholders’ expectations and needs to an acceptable level

minus

- The cost of writing, reading and maintaining this specification

Shared understanding: Enablers and obstacles

- \+ Domain knowledge
- \+ Previous joint work or collaboration
- \+ Existence of reference systems
- \+ Shared culture and values
- \+ Mutual trust
- +/– Contractual situation
- +/– Normal vs. radical design
- – Geographic distance
- – Outsourcing
- – Regulatory constraints
- – Large and/or diverse teams
- – Fluctuation

Achieving and relying on shared understanding

- Building shared understanding: The essence of requirements elicitation (cf. Chapter 7)
- Assessing shared understanding
  - Validate all explicitly specified requirements
  - Test (non-specified) implicit shared understanding
- Reducing the impact of false shared understanding
  - Short feedback cycles
  - Build and assess shared understanding early
  - Specify and validate high risk requirements explicitly

Mini-Exercise

Consider the chairlift access control case study.

(a) How can you make sure that the following explicit requirement is not misunderstood:
"The ticketing system shall provide discounted tickets which are for sale only to guests staying in one of the resort's hotels and are valid from the first to the last day of the guest's stay."

(b) We have used the term "skier" for denoting an important stakeholder role.
How can we test whether or not there is true implicit shared understanding among all people involved about what a "skier" is?

2.4 Dealing with the context

Systems cannot be understood in isolation.

- Requirements specify a system
- The system may be part of another system
- The system is embedded in a domain context
- The system boundary is not a priori clear
- The scope of a system may exceed the system boundary

Which system?

Some requirements for our sample problem:

For every turnstile, the system shall count the number of skiers passing through this turnstile.

The turnstile control software

The system shall provide effective access control to the resort's chairlifts.

Everything: equipment, computers, cards, software

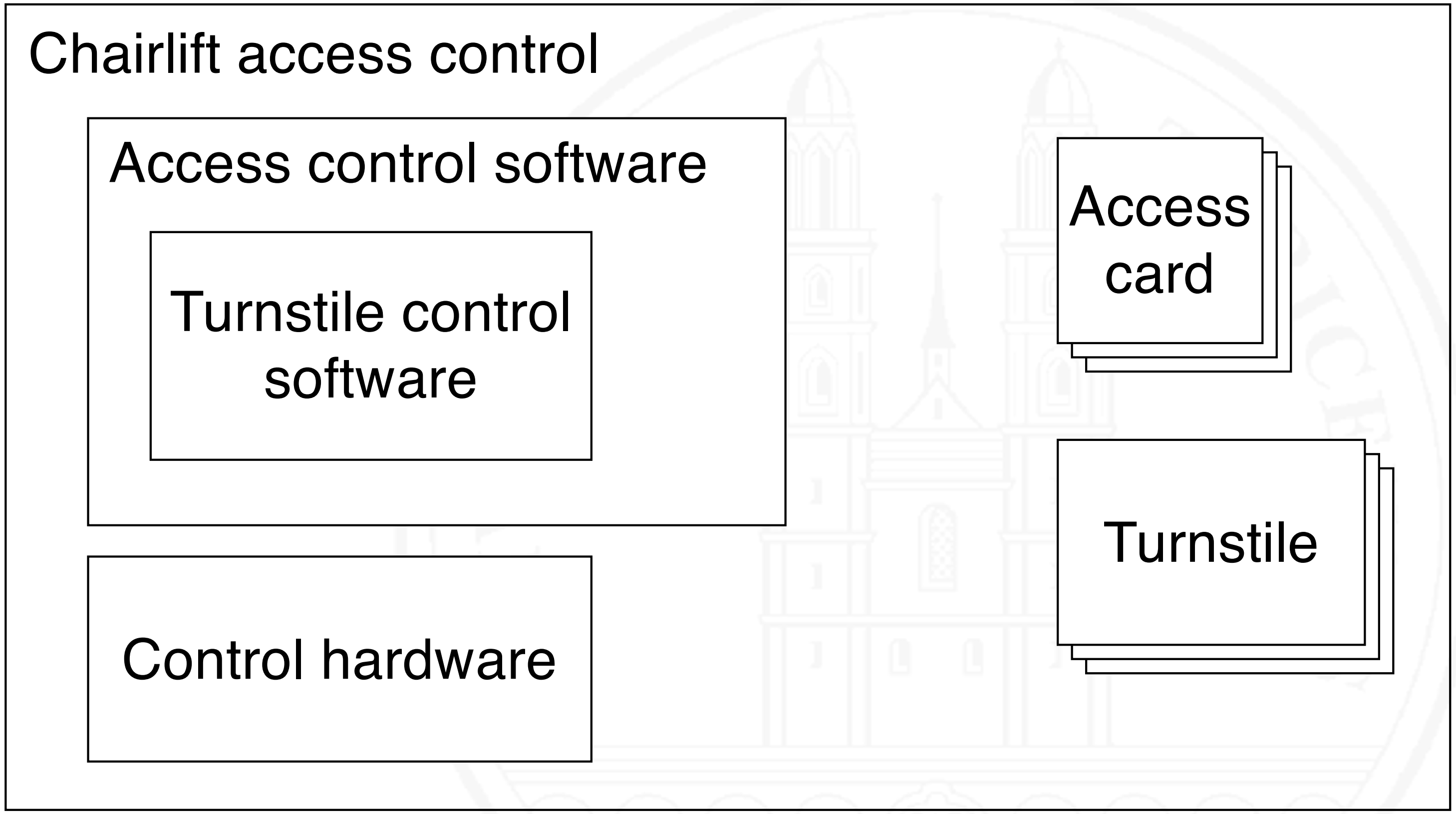
The system shall operate in a temperature range of -30° C to +30° C.

The computer hardware and the devices

The operator shall be able to run the system in three modes: normal (turnstile unlocked for one turn when a valid card is sensed), locked (all turnstiles locked), and open (all turnstiles unlocked).

The access control software for a chairlift

Systems of systems

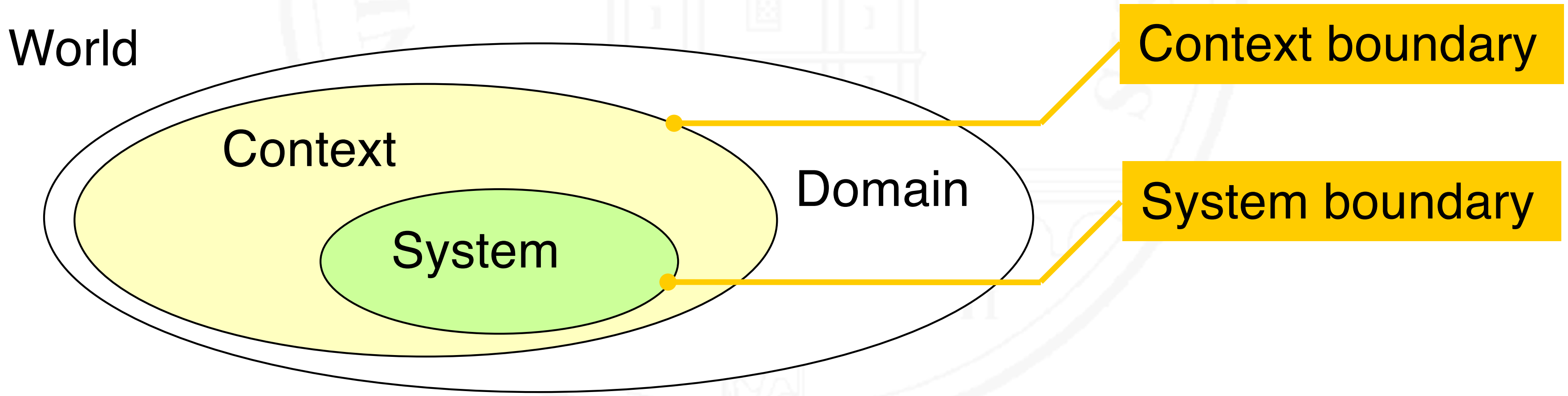


➪ Requirements need to be framed in a context

➪ Dealing with multi-level requirements is unavoidable

Context

DEFINITION. Context – 1. In general: The network of thoughts and meanings needed for understanding phenomena or utterances. 2. Especially in RE: The part of a system's environment being relevant for understanding the system and its requirements.



System boundary and context boundary

DEFINITION. System boundary – The boundary between a system and its surrounding context.

DEFINITION. Context boundary – The boundary between the context of a system and those parts of the application domain that are irrelevant for the system and its requirements.

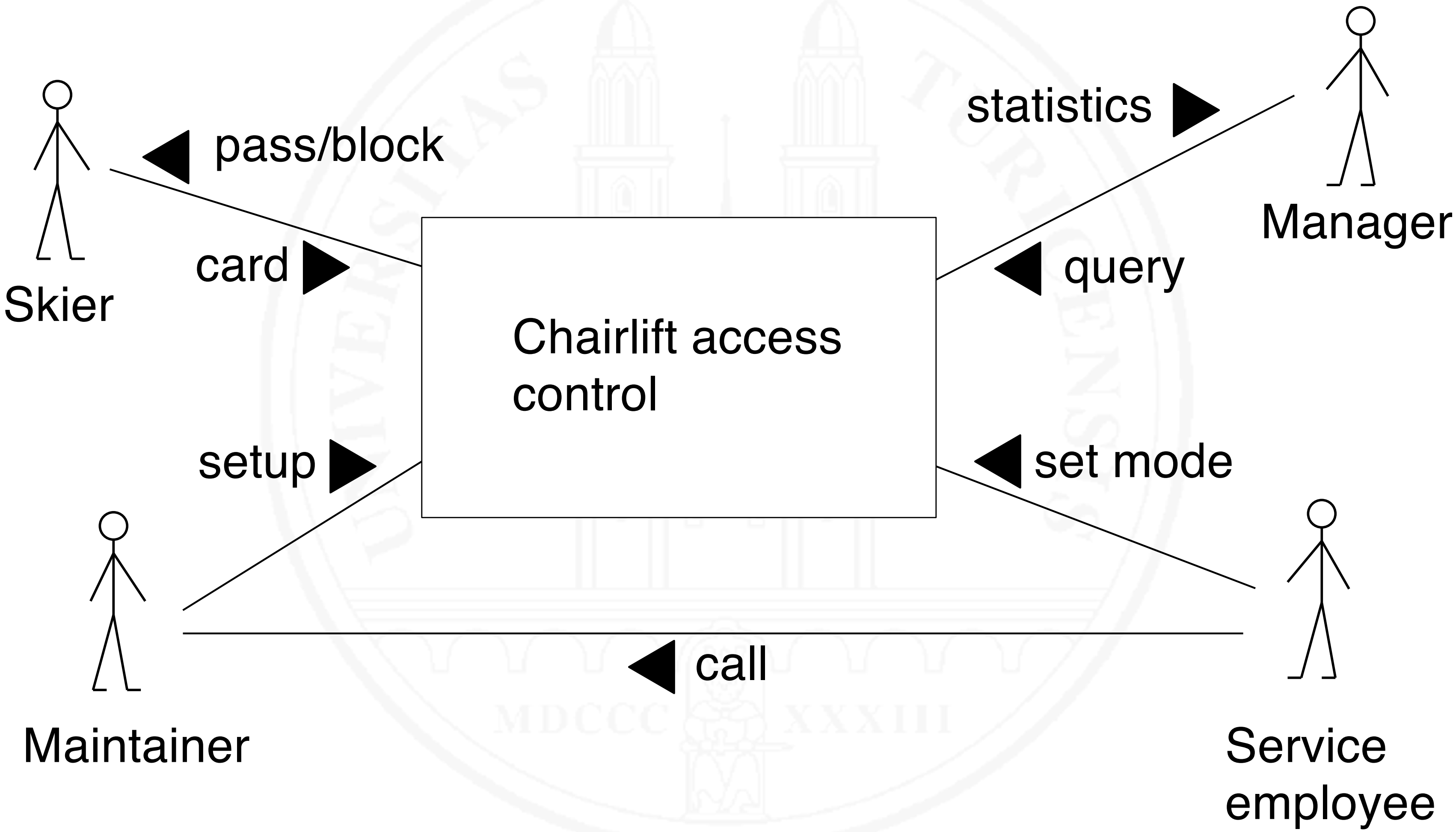
- The system boundary separates the system to be developed from its environment
- RE needs to determine the system boundary – stakeholders will have divergent views
- Information outside of the context boundary is not considered

Context models

Modeling a system in its context

- Determine the level of specification
- Usually no system internals (→ system as black box)
- Model actors which interact directly with the system
- Model interaction between the system und its actors
- Model interaction among actors
- Represent result graphically

A context diagram



Mini-Exercise

Assume that the management of the ski resort decides to voluntarily obtain a certificate from a safety certifier that the Chairlift access control system to be built is panic-safe.

The certifier might have constraints that the system must satisfy, which means that the safety certifier has to be considered a stakeholder.

Determine whether or not the context diagram (see previous slide) must be augmented, and if yes, how.

Mapping world phenomena to machine phenomena: a major RE problem

① A requirement in the world:

For every turnstile, the system shall count the number of persons passing through this turnstile.

② Mapped to a requirement for the system to be built:

The turnstile control software shall count the number of 'unlock for a single turn' commands that it issues to the controlled turnstile.

② satisfies ① only if these domain assumptions hold:

- An unlock command actually unlocks the turnstile device
- When a turnstile is unlocked, a single person passes through it
- Nobody passes through a locked turnstile (e.g. by crouching down)

The world and the machine

[Zave and Jackson 1997]
[Gunter et al. 2000]
[Jackson 2005]

Requirements must hold in the world.

But we need them to build machines (aka systems).

A machine with capabilities described by the specification S

Properties D of the domain In the real world

Required behavior R in a real world domain

The requirements problem (according to Jackson):

Given a machine *satisfying the specification S* and *assuming* that the *domain properties D hold*, the *requirements R in the world must be satisfied*: $S \wedge D \vdash R$

Mini-Exercise

Imagine the problem of two traffic lights that regulate traffic at a road construction site where only a single lane may be used. The following real-world requirement shall be satisfied:

"Ensure that, at each point in time, traffic flows at most in one direction in the one-lane region and that the control regime is both effective (actual throughput in both directions) and fair (does not favor one direction over the other)."

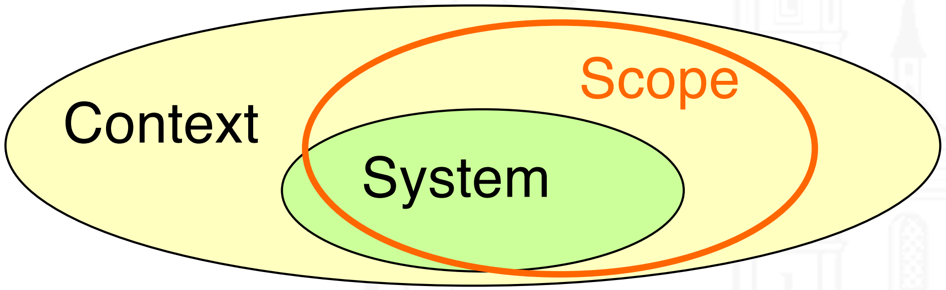
Determine

- the system requirements that the control system must meet
- which domain properties/assumptions must hold

in order to satisfy the given real-world requirement

The role of the system scope

DEFINITION. Scope (of a system development) – The range of things that can be shaped and designed when developing a system.



System scope ≠ Everything within the system boundary!

- The scope of a system may comprise parts of its context

  If this is the case, (re)-designing the context may lead to better systems than designing the system to a given context

- Some parts of a system may be given and not changeable

2.5 Requirements and design

A traditional belief:

- Requirements are about what a system ought to do
- Design deals with the problem of how to realize what has been stated in the requirements
- Requirements Engineering and System Design should be kept separate, with requirements preceding design

- Sounds good and is popular, but does not work

WHAT vs. HOW in Requirements Engineering

Is this a requirement or a design decision?

"The system prints a list of ticket purchases for a given day. Every row of this report lists (in this order) date and time of sale, ticket type, ticket price, and payment method. Every page has a footer with current date and page number."

It depends.

→ WHAT vs. HOW doesn't provide a useful distinction.

Distinguish operationally:

- If a statement is owned by stakeholders (i.e., changing it requires stakeholder approval), it's a requirement
- If a statement is owned by the supplier (i.e. the supplier may change it freely), it's part of the technical solution

Design has two facets

- Technical Design: Creating the architectural structure of a system and designing its components in detail
- Product Design: Shaping a product (or a system) with respect to its capabilities, behavior, outer form, and usage

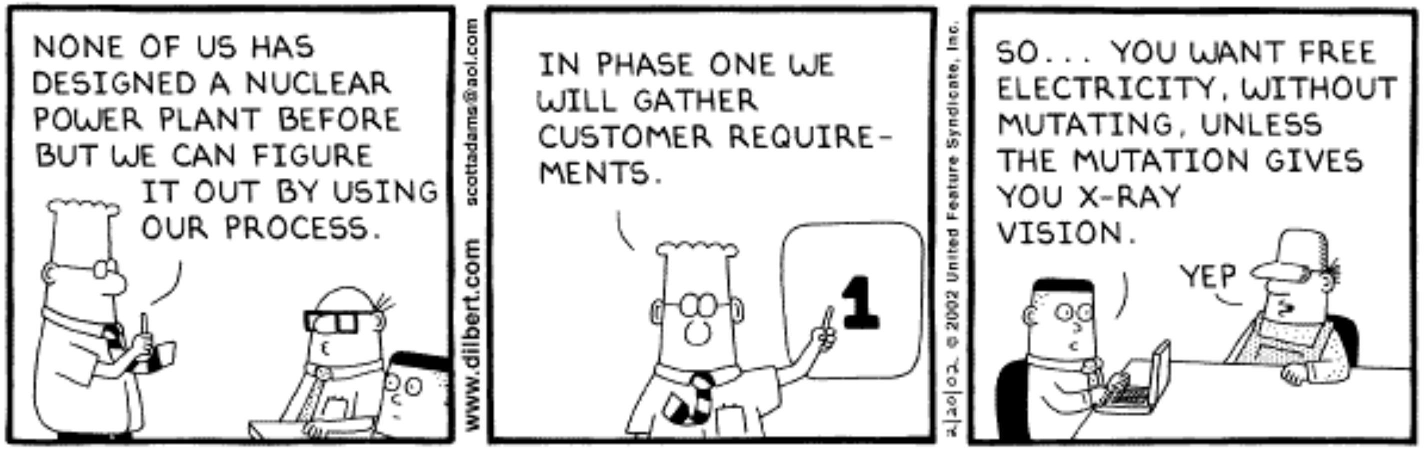
Traditional RE: Product Design comes after RE

Modern RE: Product design shapes the essence of a product
→ crucial for meeting the stakeholders' desires and needs
→ Product Design and RE are strongly intertwined

Product design for digital products is also called "Digital Design"

[Lauenroth et al. 2024]

Why care about both RE and product design?



→ We need RE competencies

→ and product design competencies

Complementary contributions

- RE contributes competencies about
  - Stakeholder identification
  - Elicitation of wishes and needs
  - Documentation of non-touchable things
  - Requirements negotiation, prioritization, and validation
- Product Design contributes competencies about
  - Usability
  - User experience design
  - Materials for physical & cyber-physical products, “digital materials” for digital products
  - Empirical product validation

Meeting requirements may not suffice to satisfy stakeholders

A requirement

The participant entry form shall have fields for the participant data *name*, *first name*, *sex*, and *person ID* and a *submit button*.

can be ruined by bad product design