Requirements Engineering I

Martin Glinz

Department of Informatics, University of Zurich
www.ifi.uzh.ch/~glinz

University of Zurich[UZH]

Department of Informatics

Requirements Engineering Research Group

Part I: Fundamentals

Part II: Requirements Engineering Practices

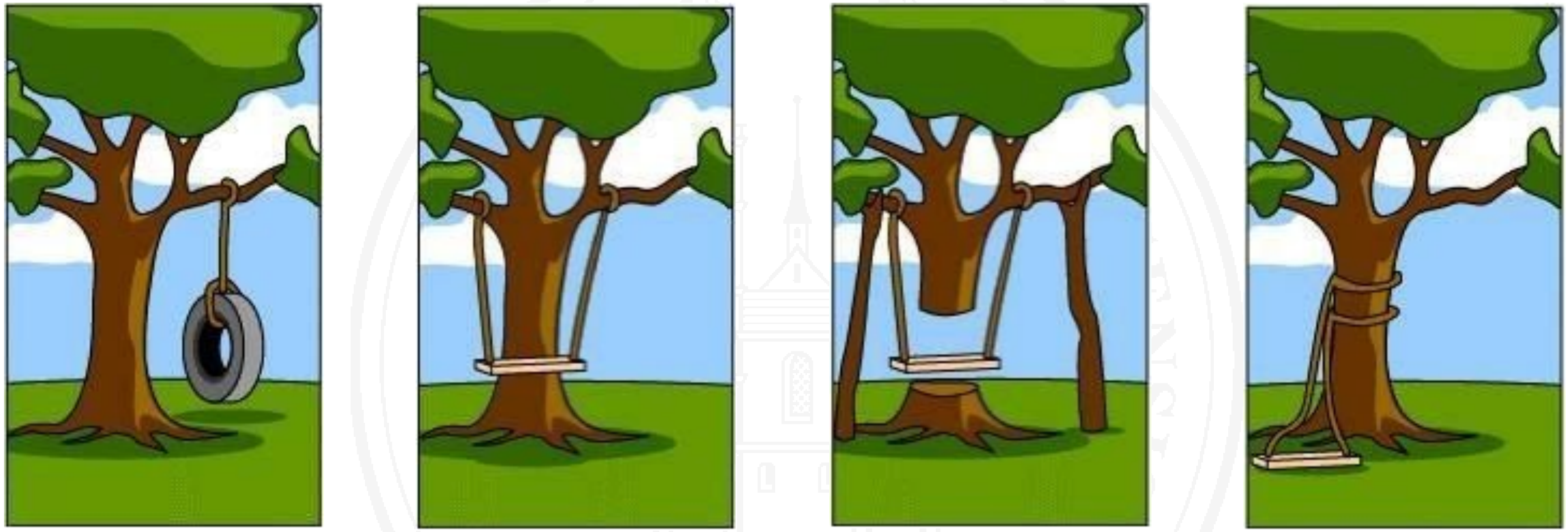
Part III: Enablers and Stumbling Blocks

Conclusions

References

1 Introduction

A communication problem



Need	Analysis	Design	Deployed System
What the customer wanted	What the analyst understood	What the architect designed	What the programmers implemented

We need to know the requirements.

DEFINITION. Requirement –

1. A need perceived by a stakeholder.
2. A capability or property that a system shall have.
3. A documented representation of a need, capability or property.

DEFINITION. Requirements Specification – A systematically represented collection of requirements, typically for a system or component, that satisfies given criteria.

[Glinz 2014] (based on IEEE 610.12-1990, slightly modernized)

Requirements specification: terminology

In some situations we distinguish between a customer (or stakeholder) requirements specification (typically written by the customer) and a system requirements specification or software requirements specification (written by the supplier).

German terminology:

- Customer/stakeholder requirements specification: Lastenheft
- System/software requirements specification: Pflichtenheft

Requirements specification may also denote the activity of specifying requirements.

Beyond requirements specifications

Agile projects frequently do not produce a comprehensive requirements specification

Instead, they express requirements in

- collections of user stories, issues, storyboards, etc.
- acceptance criteria associated with user stories
- a vision document
- implicit shared understanding among the people involved

A sample problem

A ski resort operates several chairlifts. Skiers buy RFID-equipped day access cards. Access to the lifts is controlled by RFID-enabled turnstiles. Whenever a turnstile senses a valid access card, it unlocks the turnstile for one turn, so that the skier can pass.

The task

Build a software-controlled system for managing the access of skiers to the chairlifts.

When building such a system...

- How do we determine the requirements?
- How can we analyze and document these requirements?
- How do we make sure that we've got the right requirements?
- How do we manage and evolve the requirements?

Requirements Engineering – the classic notion

DEFINITION. Requirements Engineering (RE) [Classic] – The application of a systematic, disciplined, quantifiable approach to the specification and management of requirements; that is the application of engineering to requirements.

[Adapted from the definition of Software Engineering in IEEE 610.12-1990]

Metaphor: upfront engineering

Goal: complete, unambiguous requirements prior to design

Smells: paper, process

Reality check: Does this always work?

Wait a minute – it's about customers' needs

DEFINITION. Requirements Engineering [Customer-oriented] – Understanding and documenting the customers' desires and needs.

[Glinz 2004, Chapter 7, inspired by Gause and Weinberg (1989)]

Metaphor: Customer satisfaction

Goal: Understand the customer

Reality check:

(1) Why not just code what the customer desires and needs?

(2) Who is "the customer"?

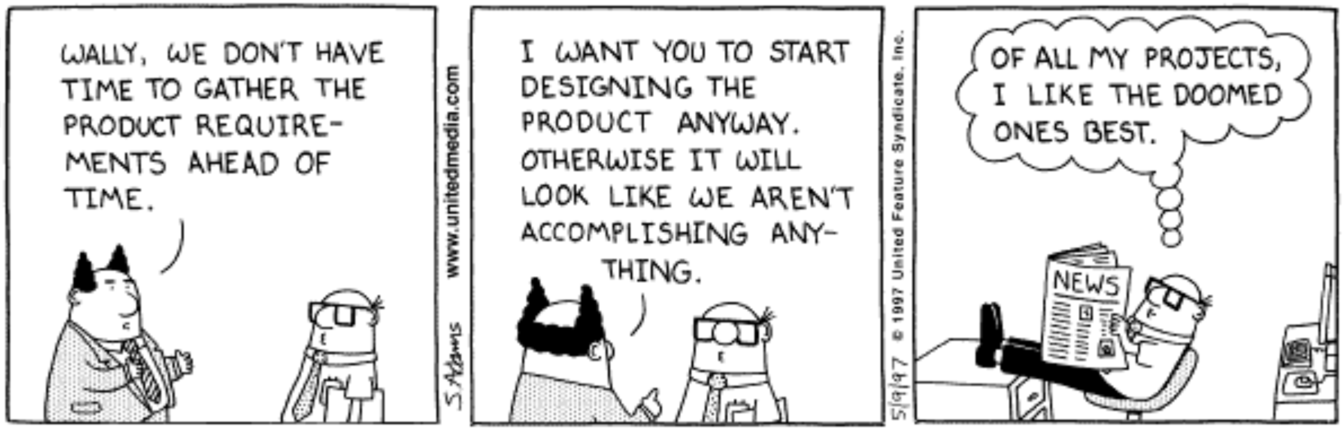
Where's the value?

DEFINITION. Requirements Engineering [Risk-oriented] – Specifying and managing requirements to minimize the risk of delivering a system that does not meet the stakeholders' desires and needs.

[Glinz (2014) based on my work on requirements risk]

Metaphor: Balancing effort and value

Goal: Mitigate risk



Risk-based RE

"We have no time for a complete specification."

"This is too expensive!"

"We're agile, so rough stories suffice."

➪ Wrong approach

Right question: "How much RE do we need such that the risk of deploying the wrong system becomes acceptable?"

Rule:

The *effort* spent for Requirements Engineering shall be *inversely proportional* to the *risk* that one is willing to take.

A synoptic definition of RE

[Glinz (2014); for the definition of 'stakeholder' see Chapter 2]

DEFINITION. Requirements Engineering – A systematic and disciplined approach to the specification and management of requirements with the following goals:

(1) Knowing the relevant requirements, achieving a consensus among the stakeholders about these requirements, documenting them according to given standards, and managing them systematically,

(2) Understanding and documenting the stakeholders' desires and needs,

(3) Specifying and managing requirements to minimize the risk of delivering a system that does not meet the stakeholders' desires and needs.

A note on terminology

- Lots of sources for today's terminology
  - Textbooks and articles about RE
  - IEEE 610.12 (1990) – a slightly aged glossary of software engineering terminology
  - IEEE 830-1998 – an outdated, but still cited RE standard
  - ISO/IEC/IEEE 29148 (2011) – a new, but still rather unknown RE standard; provides definitions of selected terms, some of them being rather uncommon
  - IREB Glossary [Glinz 2014] – influential through IREB's certification activities; used as a terminology basis in this course

Why specify requirements?

- Lower cost
  - Reduce error cost
  - Reduce rework cost
- Manage risk
  - Meet stakeholders' desires and needs
  - Reliable estimates for deadlines and cost

Supplier makes profit

Customer is satisfied

☞ The economic effects of Requirements Engineering are (almost ever) indirect ones; RE as such just costs!

2 Principles of Requirements Engineering

Nine basic principles

1. Stakeholders
2. Systems and context
3. Problems, requirements and solutions
4. Value-orientation
5. Shared understanding
6. Validation
7. Evolution
8. Innovation
9. Systematic and disciplined work

2.1 Stakeholders

Who is “the customer”?

In our sample problem: Just the skiers?

In reality: Many persons in many roles are involved

DEFINITION. Stakeholder – A person or organization that has a (direct or indirect) influence on a system’s requirements.

Indirect influence also includes situations where a person or organization is impacted by the system.

[Glinz and Wieringa 2007]
[Macaulay 1993]

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

2.2 Systems and context

Requirements never come in isolation.

- ❍ Requirements specify a system
- ❍ The system may be part of another system
- ❍ The system is embedded in a domain context
- ❍ 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

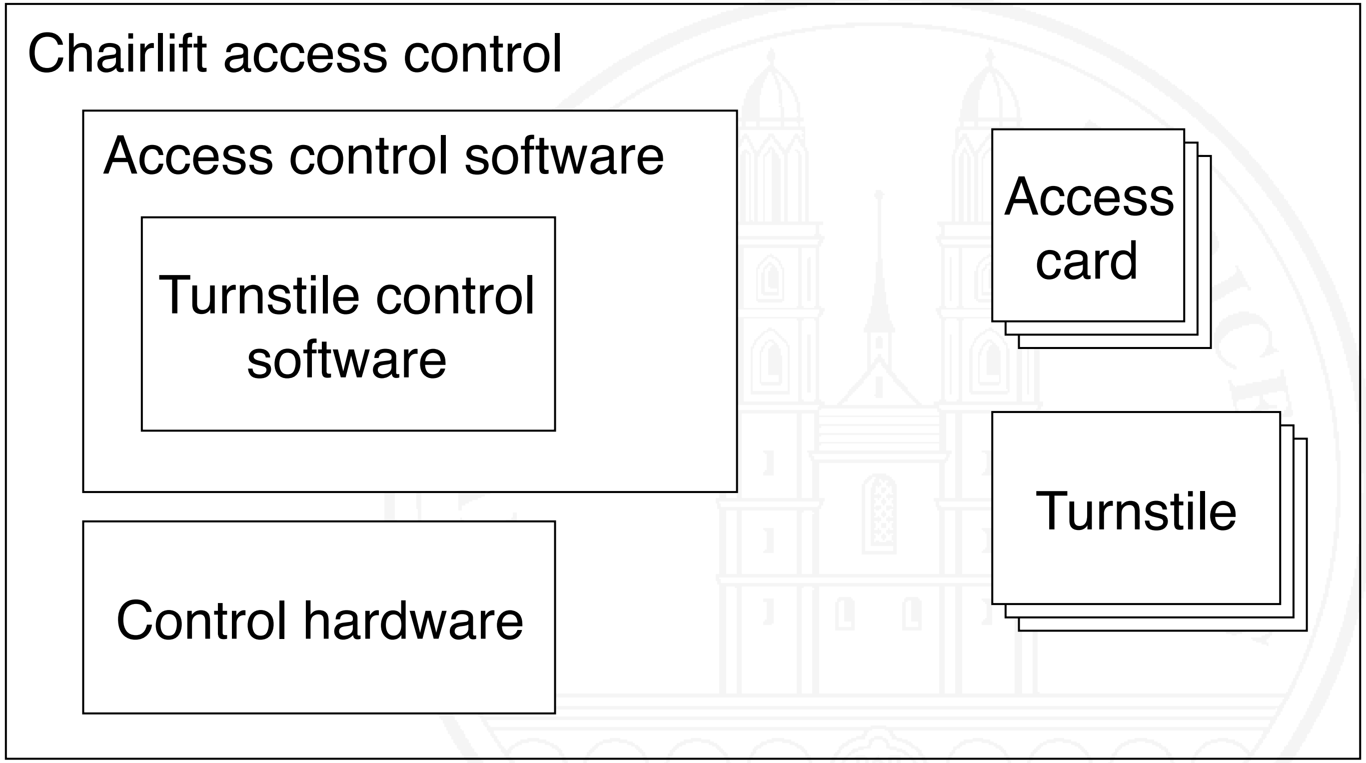
The system shall operate in a temperature range of -30° C to +30° C.

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

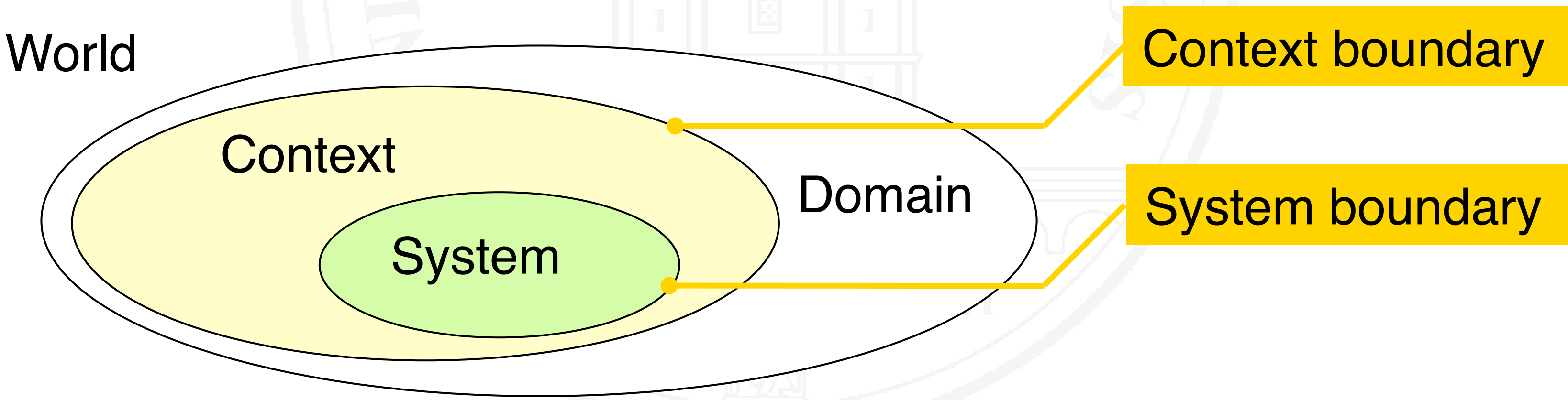


➪ Requirements need to be framed in a context

➪ 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 – Boundary between the context of a system and those parts of the application domain that are irrelevant for the system and its requirements.

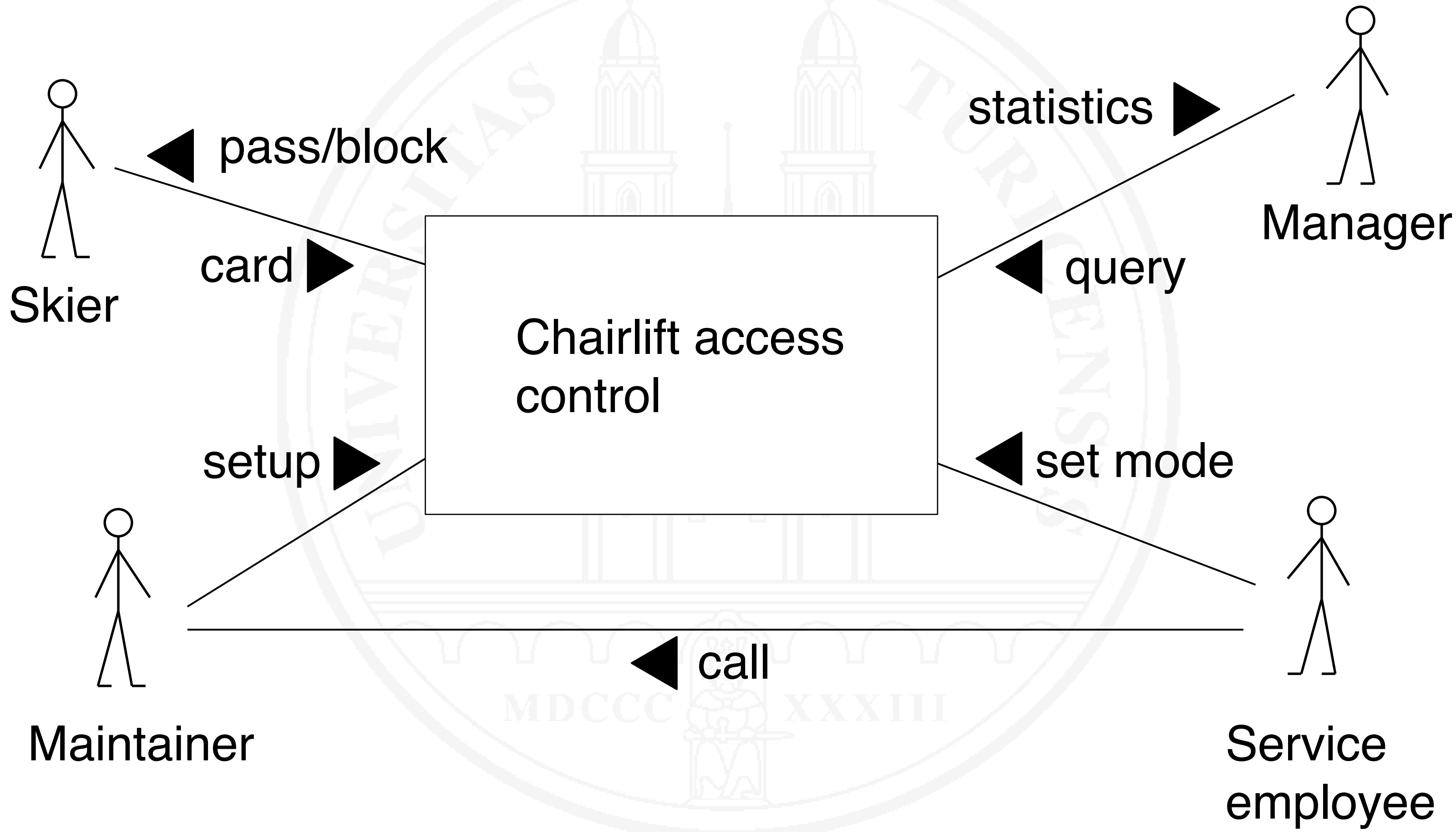
- The system boundary separates the system to be developed from its environment
- RE needs to determine the system boundary
- Information outside of the context boundary is not considered

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



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]
[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

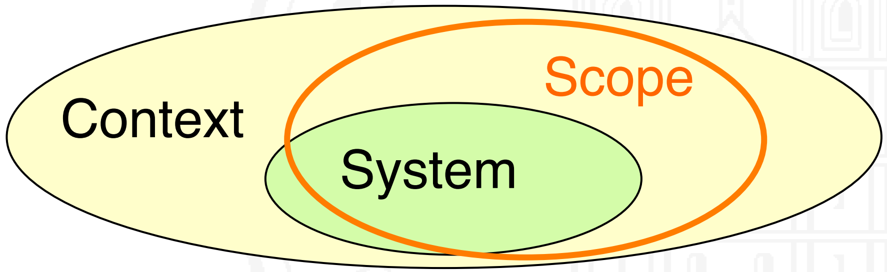
- the system requirements that the control system must meet
- which domain properties/assumptions must hold

in order to satisfy the given real-world requirement

The role of the system scope

DEFINITION. Scope (of a system) – 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

2.3 Problems, requirements and solutions

[Swartout and Balzer 1982]

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

Traditional Requirements Engineering: the waterfall

- Start with a complete specification of requirements
- Then proceed to desiging 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

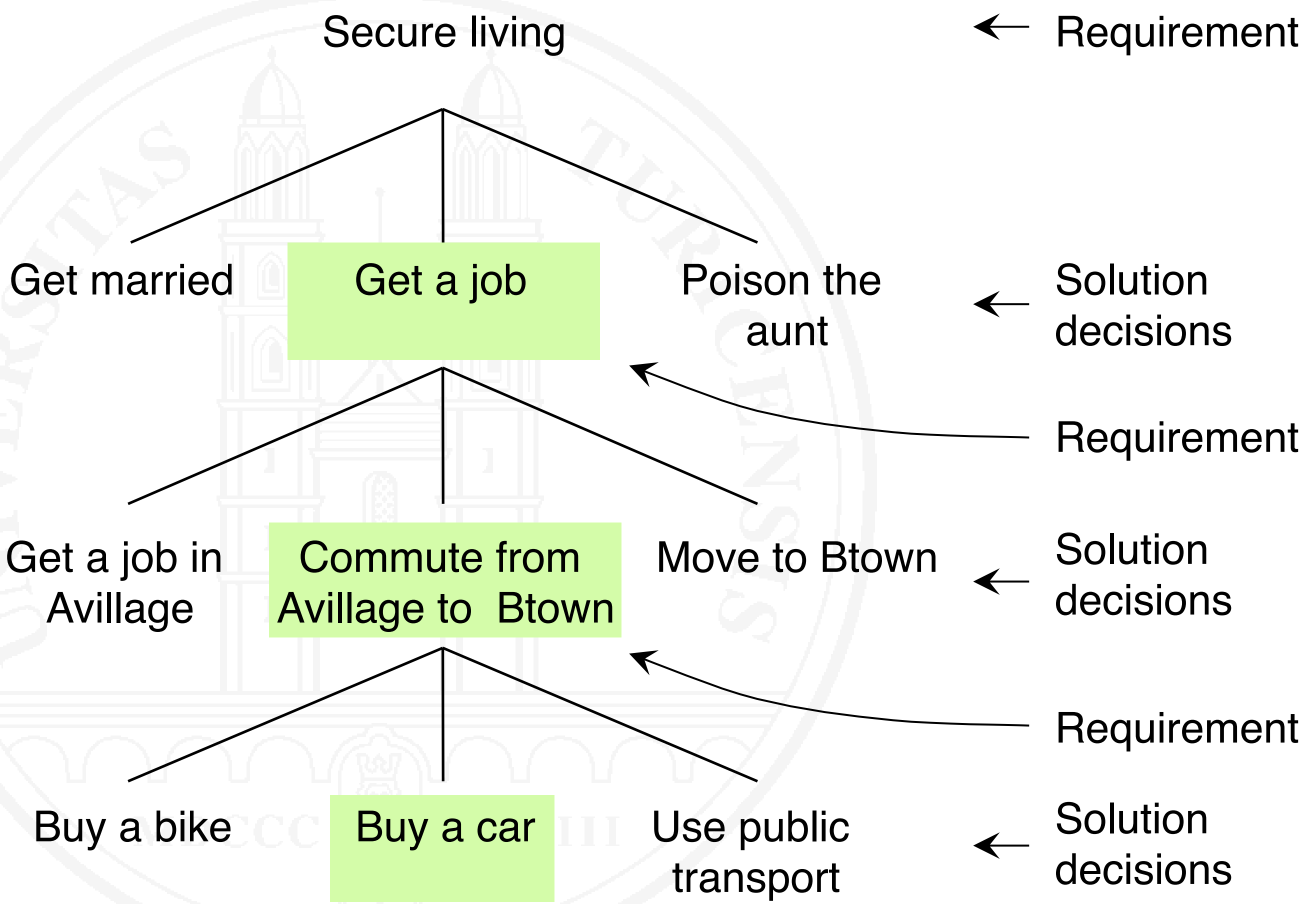
Lower level requirements

➪ 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.



Typical requirement layers

Using a railway system as an example

- ☆ Business: “More people than today shall be transported using the existing tracks.”
- ☆ System: “The minimal distance between two trains shall always be greater than the current maximum braking distance of the successive train.”
- ☆ Software: “The current maximum braking distance shall be computed every 100 ms.”

WHAT vs. HOW in Requirements Engineering

Traditional belief: WHAT = Specification, HOW = Design

But: is this a requirement or a solution 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.”

➜ WHAT vs. HOW is context-dependent and doesn’t provide a useful distinction.

Distinguishing requirements and solutions

- WHAT vs. HOW doesn't work
- Requirements and solutions should be documented separately
- 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 solution

2.4 Value-orientation

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]

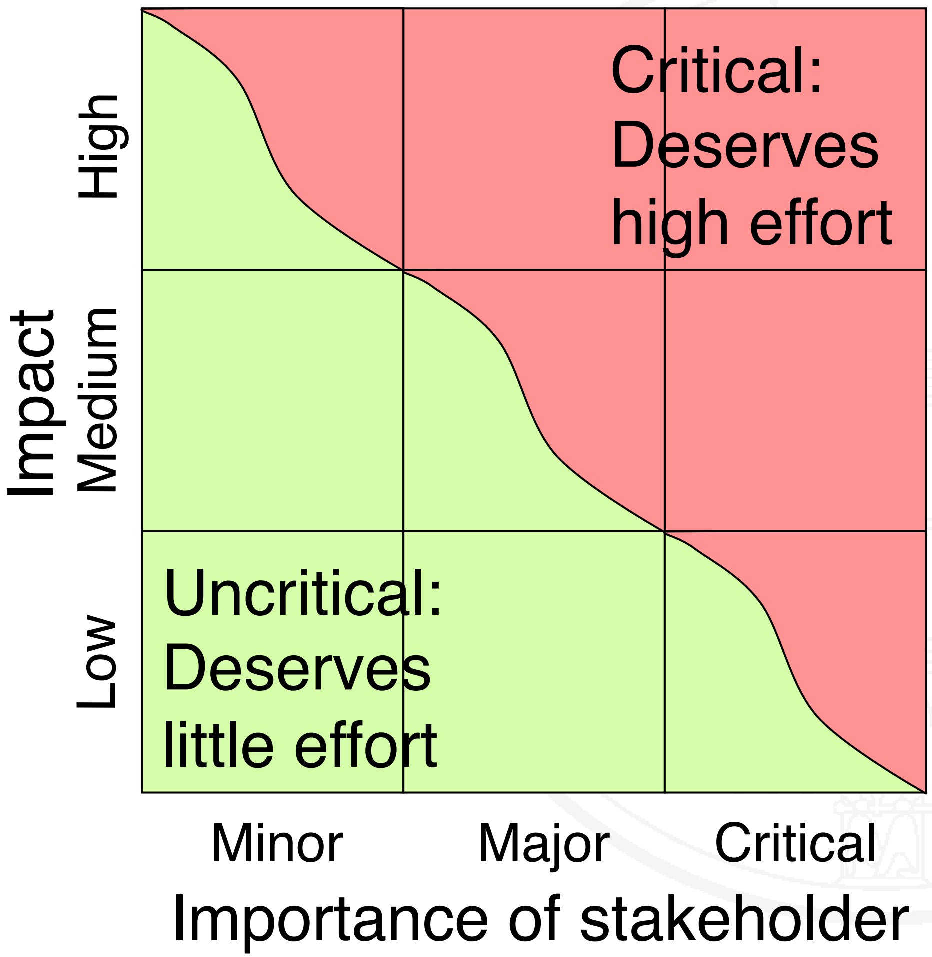
- Requirements shall deliver value
- Value of a requirement:
  - The benefit of reducing development risk (i.e. the risk of not meeting the stakeholders' desires and needs)
  - minus the cost of specifying the requirement

☞ 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



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

[Glinz 2008]

Assessing risk: other factors

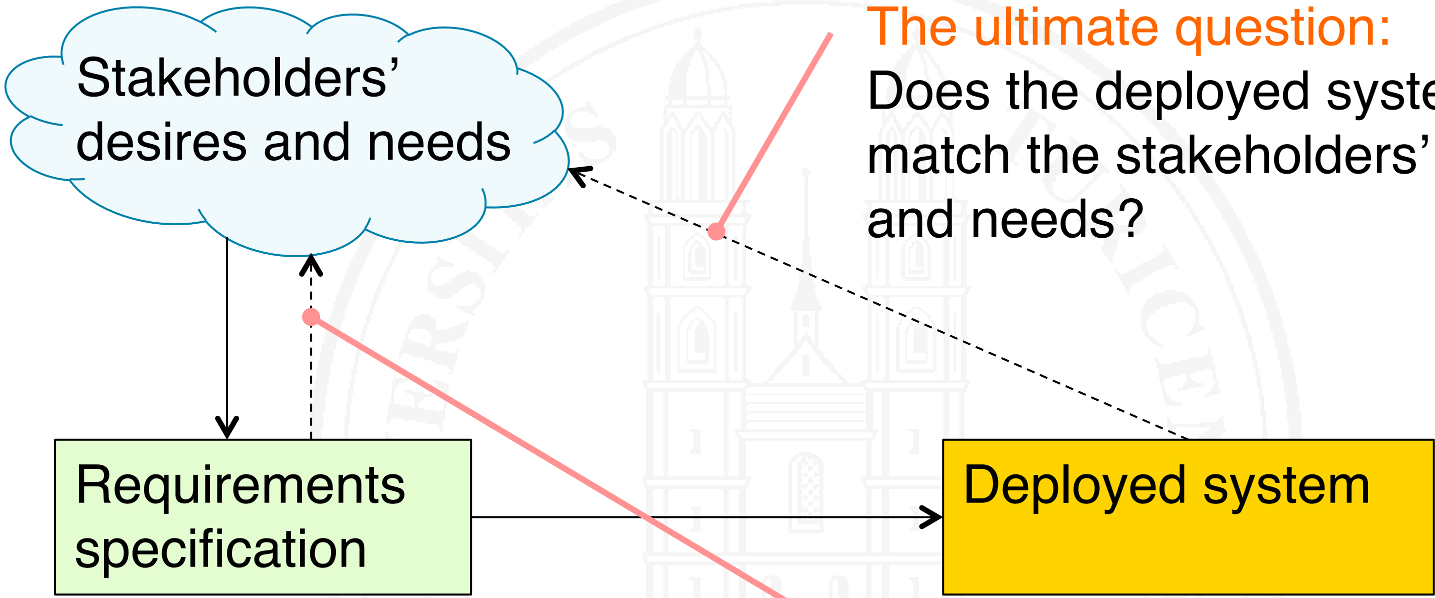
- ❍ Specification effort
- ❍ Distinctiveness
- ❍ Shared understanding
- ❍ Reference systems
- ❍ Length of feedback-cycle
- ❍ Kind of customer-supplier relationship
- ❍ Certification required

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

2.5 Shared understanding

- A basic prerequisite for any successful development of systems
- Created, fostered and assured in Requirements Engineering
- For more, see Chapter 4

2.6 Validation



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

2.7 Evolution

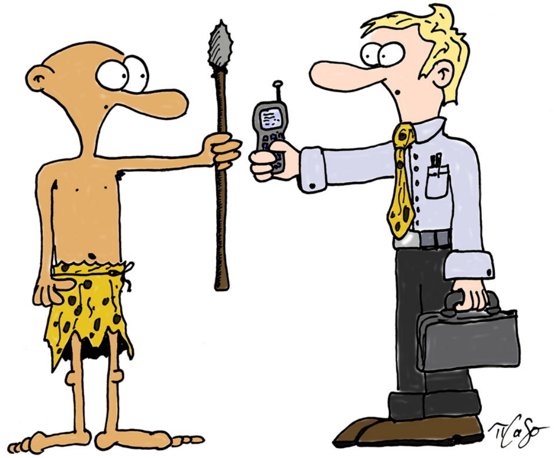


Image © C. Sommer /EKHN

The world evolves.

So do requirements.

The problem:

Keeping requirements stable...

... while permitting requirements to change

Potential solutions

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

2.8 Innovation

Image © Apple

"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.

More of the same will not excite anybody.

Strive for making stakeholders happy.

Innovative requirements are the key.

2.9 Systematic and disciplined work

We can't do without.

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

3 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

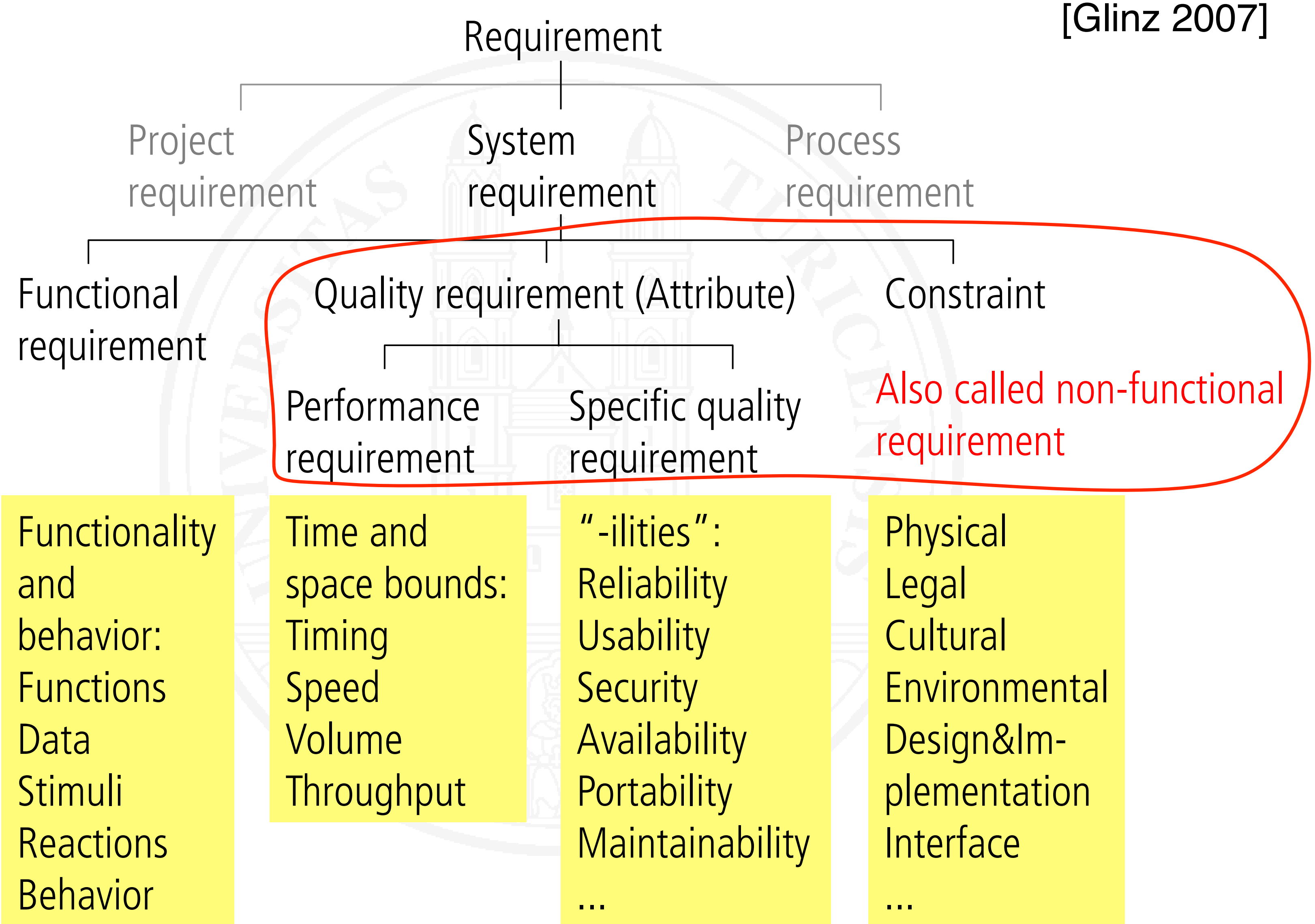
Requirements have a concern

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

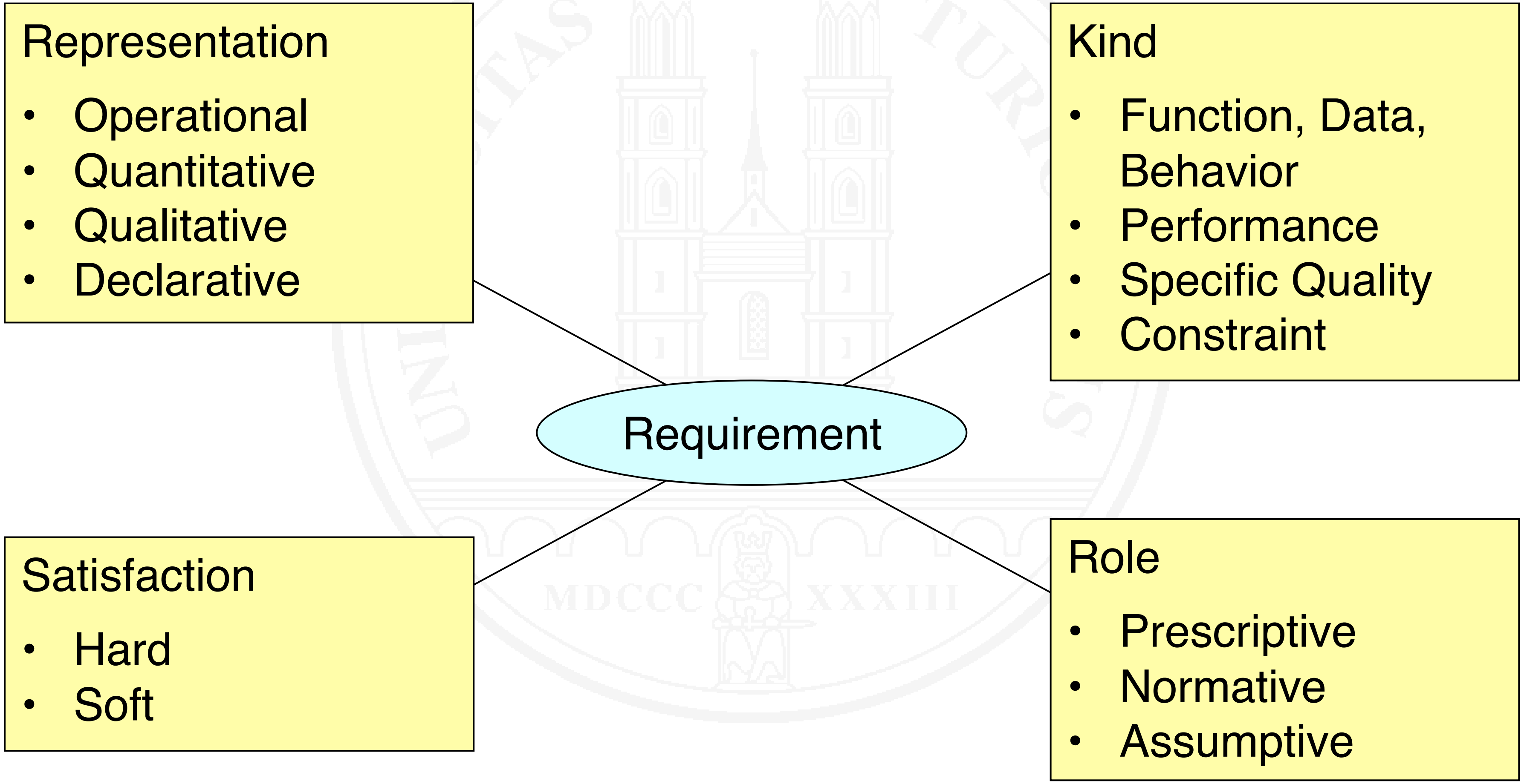
Classification according to kind

[Glinz 2007]



Beyond kind: A faceted classification

[Glinz 2005, 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**

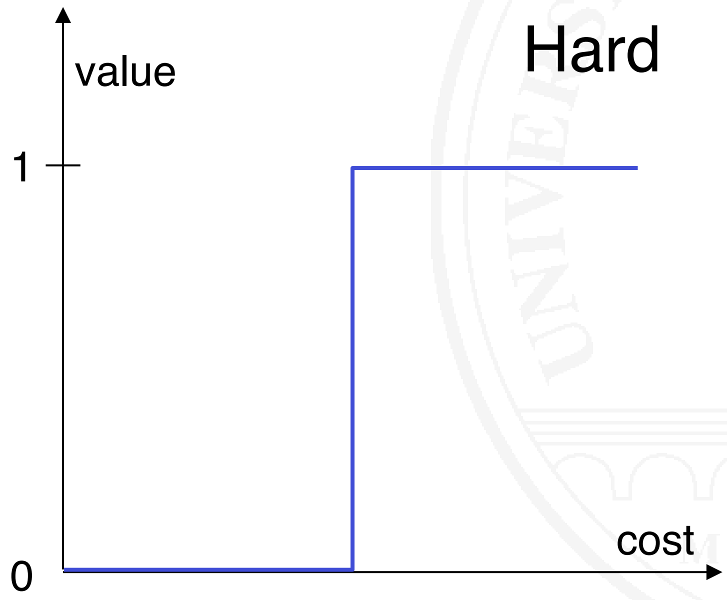
The turnstile control software shall count the number of ‘unlock for a single turn’ commands that it issues to the controlled turnstile. **Operational**

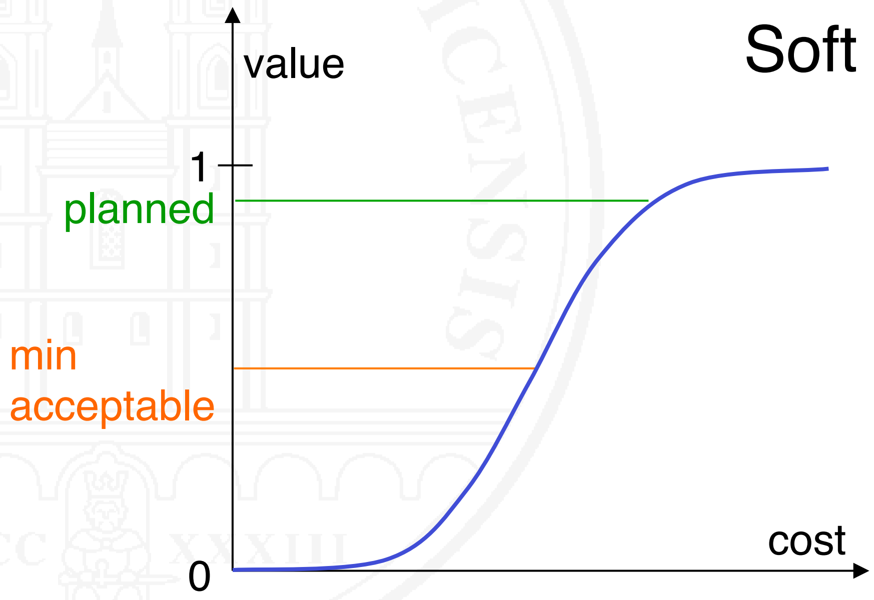
Representation informs validation

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

4 Shared understanding

[Glinz and Fricker 2013, 2015]

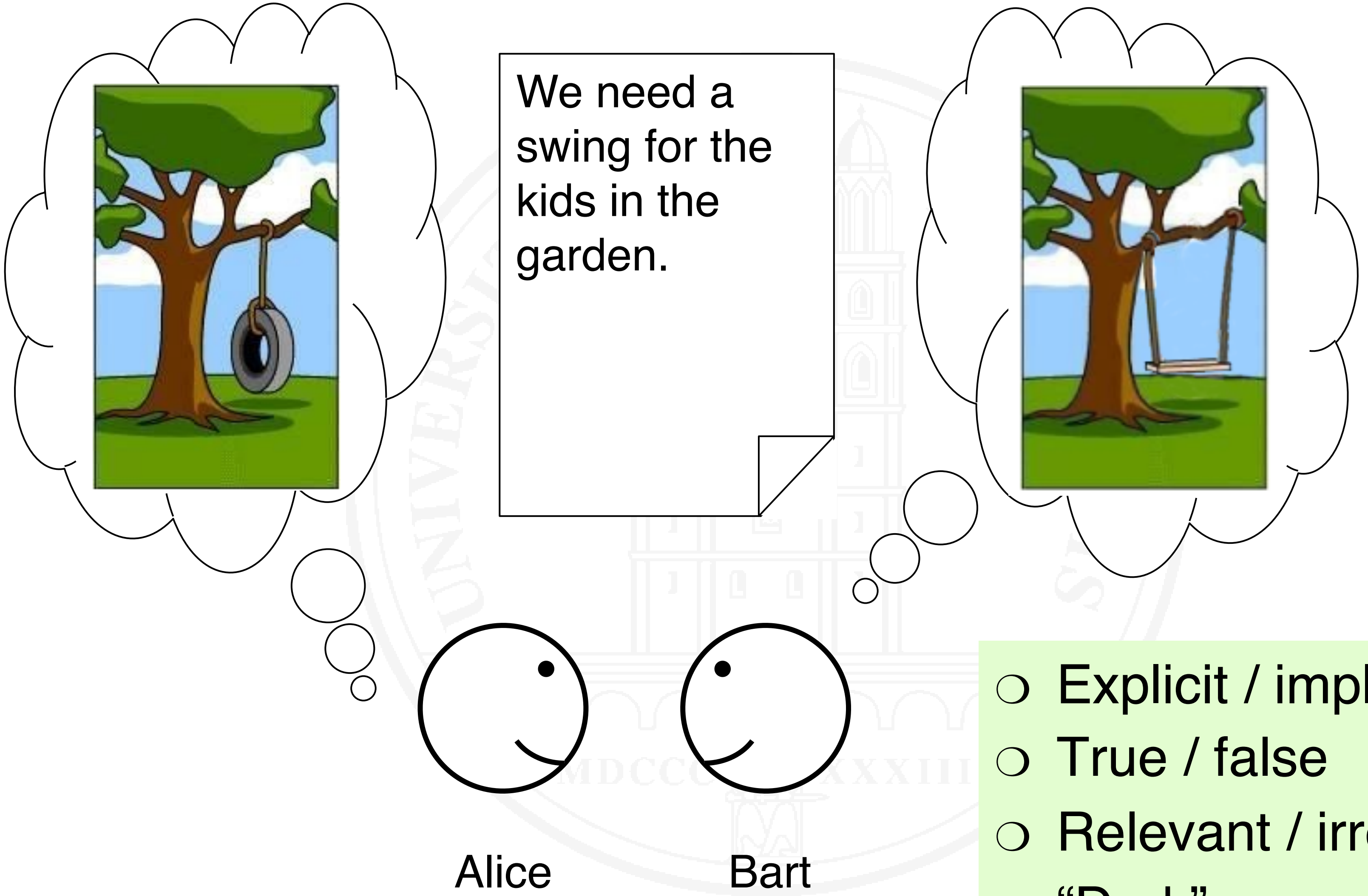
Two disturbing observations:

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

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

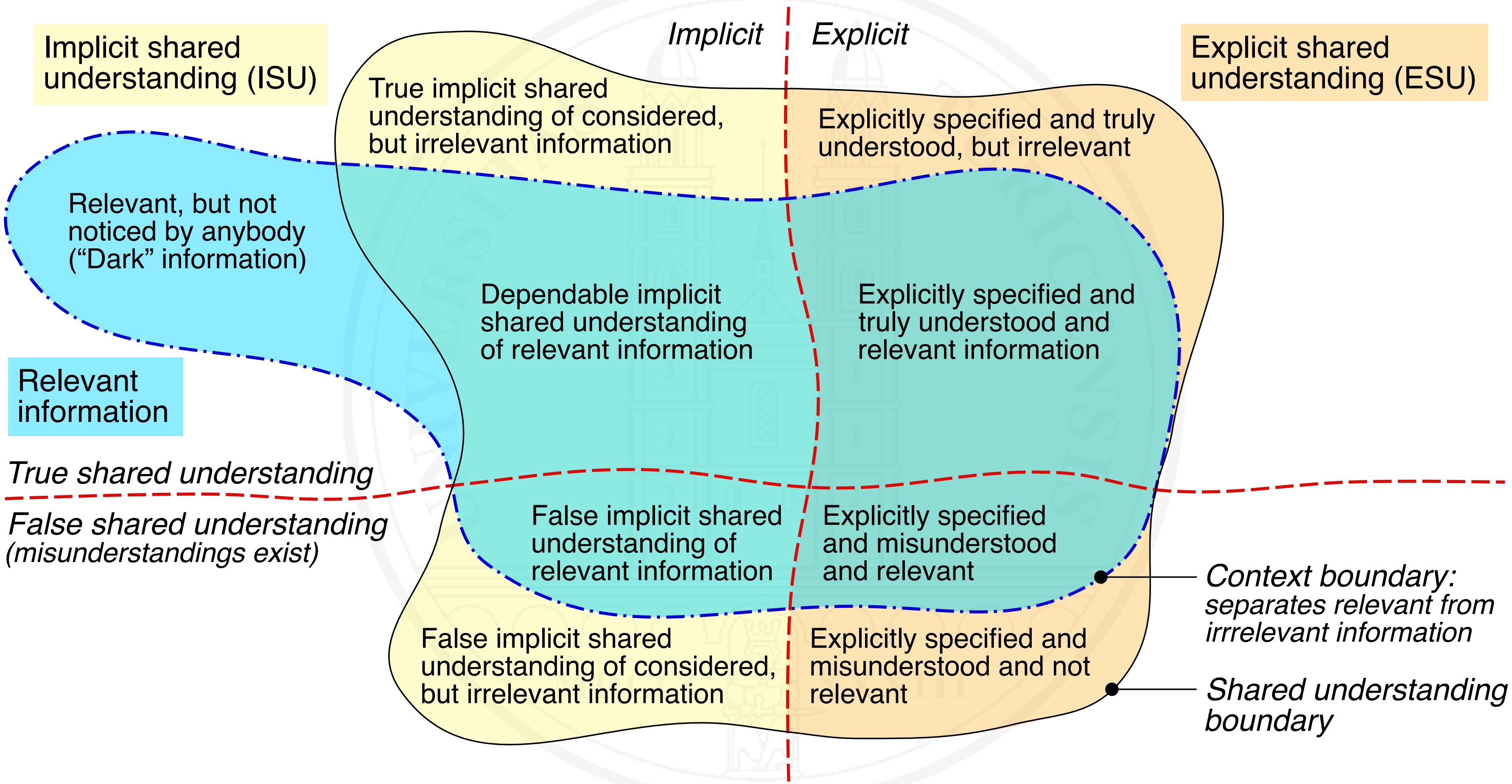
- How do we establish shared understanding?
- How can we rely on shared understanding?

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

In fact a value problem (cf. Principle 4 in Chapter 2)

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?