

Requirements Engineering

Lecture 1

Prof. Dr. Alexander Pretschner

Chair of Software & Systems Engineering
TUM Department of Informatics
Technical University of Munich

Orientation



Recap:

- Organizational issues

Coming up:

- What does Requirements Engineering consist of?
- Definition of terms
- Why is it so hard?

What is Requirements Engineering not?



It is an **engineering** process!

"The creative application of scientific principles to design or develop structures [...]"

- Wikipedia

„Corona-Warn-App“

The German „Corona-Warn-App“ detects when two smartphones, which are both switched on and have both Bluetooth as well as location services activated, are in close proximity to each other for more than 15 minutes.

Which **essential aspect** of a software system is missing here?

Let's analyze:

Which mistakes have been made and by whom?

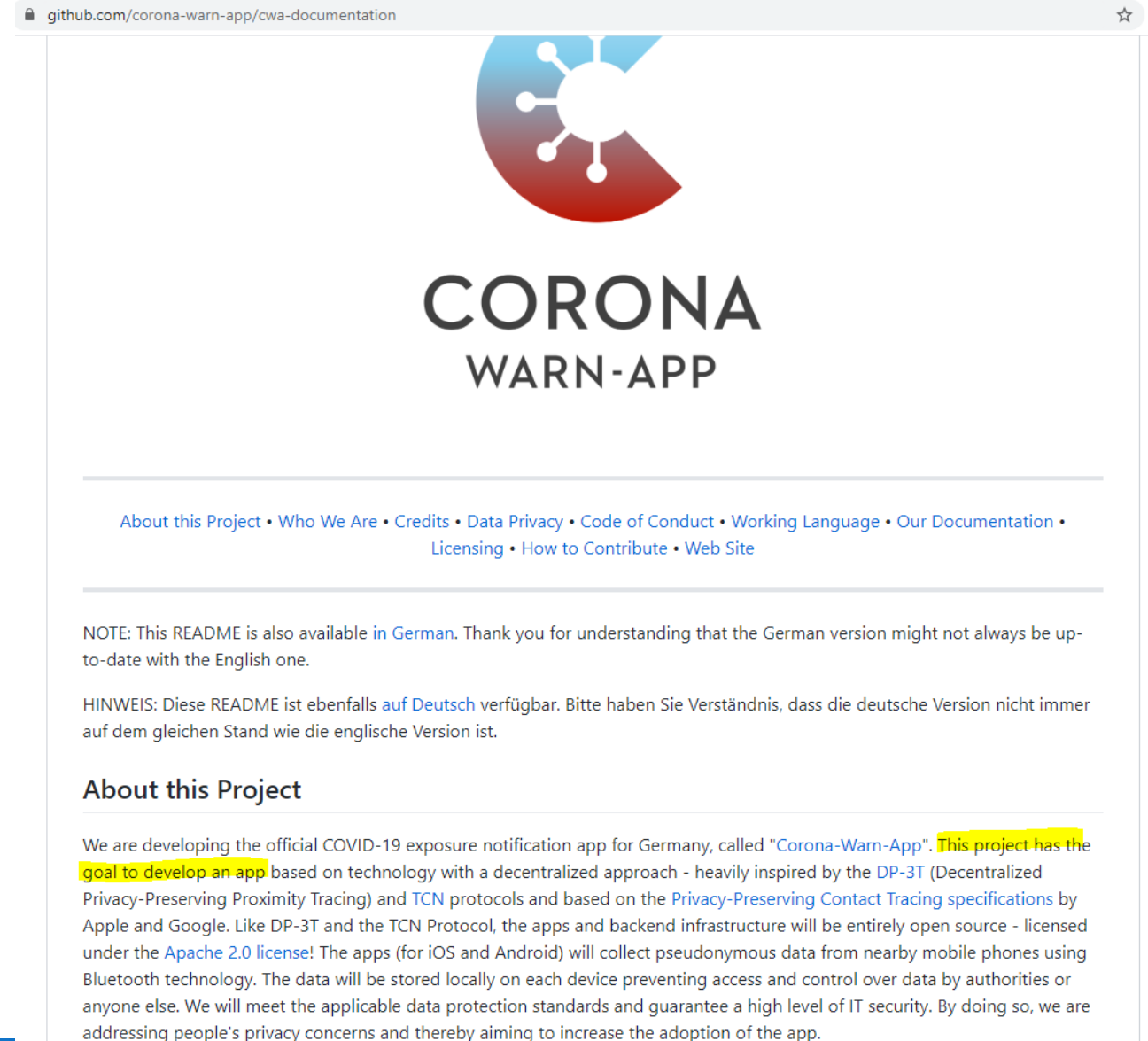
*Was the German „Corona-Warn-App“ a **Requirements Engineering** failure?*



Goal of „Corona-Warn-App“

“This project has the goal to develop an app based on technology with a decentralized approach - heavily inspired by the DP-3T and TCN protocols and based on the Privacy-Preserving Contact Tracing specifications by Apple and Google.”

This is not the goal of the project.



The screenshot shows the GitHub repository page for the Corona-Warn-App documentation. At the top, the URL is `github.com/corona-warn-app/cwa-documentation`. The main header features the Corona-Warn-App logo, which consists of a stylized virus icon with a red-to-blue gradient and the text "CORONA WARN-APP" below it. A navigation bar contains links: "About this Project", "Who We Are", "Credits", "Data Privacy", "Code of Conduct", "Working Language", "Our Documentation", "Licensing", "How to contribute", and "Web Site". Below this, a note states: "NOTE: This README is also available [in German](#). Thank you for understanding that the German version might not always be up-to-date with the English one." This is followed by a German version of the note: "HINWEIS: Diese README ist ebenfalls [auf Deutsch](#) verfügbar. Bitte haben Sie Verständnis, dass die deutsche Version nicht immer auf dem gleichen Stand wie die englische Version ist." The section "About this Project" follows, containing the text: "We are developing the official COVID-19 exposure notification app for Germany, called 'Corona-Warn-App'. **This project has the goal to develop an app** based on technology with a decentralized approach - heavily inspired by the DP-3T (Decentralized Privacy-Preserving Proximity Tracing) and TCN protocols and based on the [Privacy-Preserving Contact Tracing specifications](#) by Apple and Google. Like DP-3T and the TCN Protocol, the apps and backend infrastructure will be entirely open source - licensed under the [Apache 2.0 license](#)! The apps (for iOS and Android) will collect pseudonymous data from nearby mobile phones using Bluetooth technology. The data will be stored locally on each device preventing access and control over data by authorities or anyone else. We will meet the applicable data protection standards and guarantee a high level of IT security. By doing so, we are addressing people's privacy concerns and thereby aiming to increase the adoption of the app."

„Corona-Warn-App“ done right

Consider what goals motivate a "Corona Warn App".

- Saving lives
- Containment of the virus
- Avoiding situations where many infections occur
- Notification of potentially infected individuals
- ...

How can we achieve these goals?

How would you specify a system to achieve these goals?

After this lecture, you will know!



Designing „Rent-a-Scooter“



How do we specify that system?

Where do we start?

Discussion: Designing „Rent-a-Scooter“



Who are relevant stakeholders for this project?

How many can we collect?

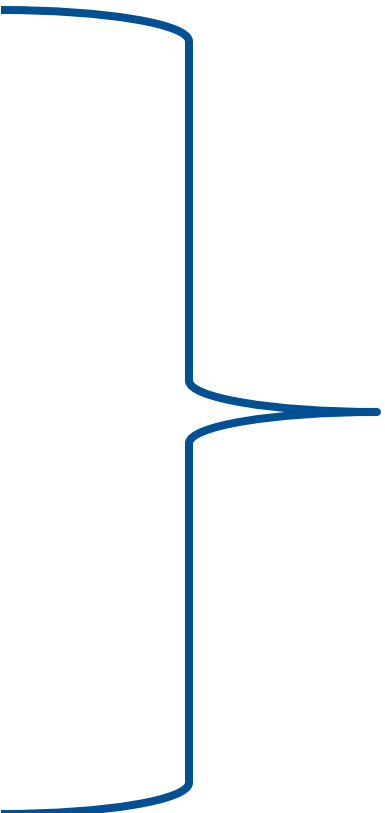
In this lecture, you will learn to..

- Identify the stakeholders and analyze their relationships
- Identify and resolve conflicts before they materialize in the project
- Select appropriate elicitation techniques
- Use viewpoints, abstraction levels and system models
- Understand advantages and disadvantages of formalization
- Apply basics of ethical deliberation
- Break down subjective non-functional requirements
- Establish automatic quality control for requirements
- Handle change in projects
- [...]

What is RE? Related terms

Analysis phase
System analysis
Requirement analysis
Requirements engineering
Requirements development
Requirement phase / definition phase
Specification phase
Requirement definition/specification
System Specification
Requirements Management
...

Design Thinking
Usability Engineering



All these terms describe
concepts that are summarized
by the generic term
Requirements Engineering

Definition Requirements Engineering (RE)

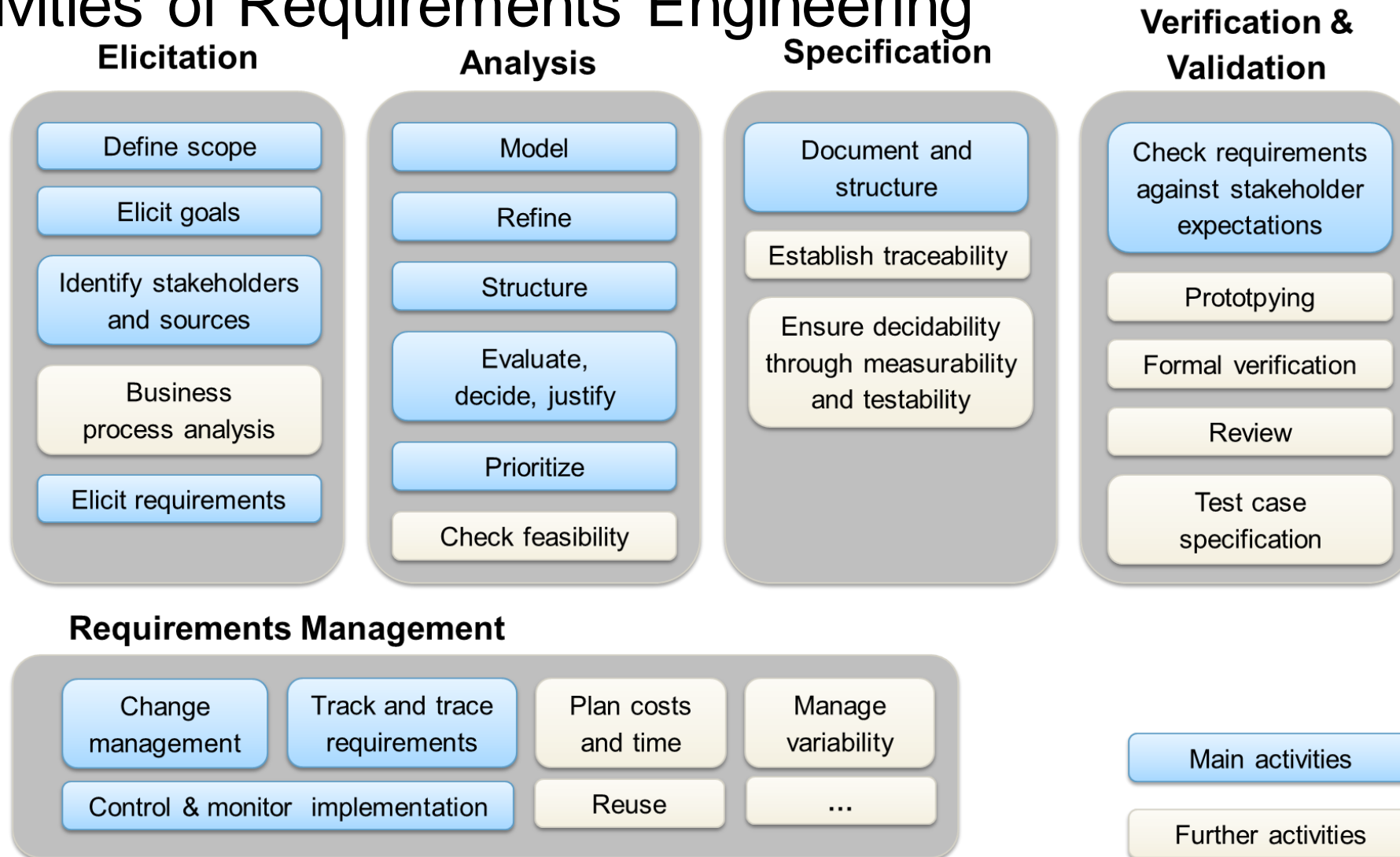
Requirements Engineering (RE) is an iterative, systematic approach, designed for efficiency and effectiveness, with the goal of creating explicit **requirements and system specifications** that are agreed upon with all stakeholders.

Level of detail depends on context: RE for a shell script? A scheduler? A web app? Rent-a-scooter?

Core Activities:

- Collect/identify requirements (**Elicitation**)
- Analyze, negotiate, consolidate requirements (**Analysis**)
- Structure, model, document requirements (**Specification**)
- Check requirements for quality and validity (**Validation & Verification**)

Core activities of Requirements Engineering



Definition Requirements Management (RM)

Requirements Management (RM) aims at an efficient and effective administration and use of requirements throughout the entire system life cycle, including

- Archiving
- Modification based on new findings during development
- Tracking, tracing and verification of requirements:
 - Impact analyses
 - Support of change processes

→RM often involves a merger of RE activities

Example: Requirements and project management:

Cost and risk assessment, planning, etc.

Project management: incremental development, timeboxing, ...

Definition: Requirement

A **requirement** is:

1. a condition/capability/feature that a **stakeholder** demands for a product or process to solve a problem or achieve a goal.
Most important stakeholder: The user! (Remember there are users and operators.)
2. a condition/capability/feature that a **system** must meet in order to fulfill a contract, standard, specification or other formal document.
3. a documented representation of a condition/ability/property as defined in 1. or 2.

We distinguish between **user requirements** and **system requirements**

Requirements: a first classification

More details in Lecture 7

Functional requirements

- Requirements that describe the **behavior** of the system, especially with regard to its use (including time behavior when relevant for functionality: mp4 player)

More details in Lecture 8

Quality requirements

- Requirements on quality **properties** of the system (characteristic/quantitative properties with regard to behavior), general conditions

Process requirements (project-specific requirements):

- Development process requirements (project plan, milestones, budget, deadlines, quality assurance, etc.)
- Implementation specifications (boundary conditions, constraints): Components or technologies to be used

Quality and process requirements are often vaguely called **non-functional requirements**.

Requirements: a second classification

Business requirements

- Describe the benefits and value that is created by this project
- Mostly very high-level
- Example: *“We want to achieve data integrity for the data created by department x.”*

Stakeholder requirements / User requirements

- Describe what stakeholders / users need and want to do with the system
- User requirements are used as input for deducing system requirements.
- Example: *“As a user, I want to be able to restore my data if the system crashed.”*

System requirements:

- Often used as input for implementation
- Example: *“The system shall create backups of the data every 24 hours.”*

Requirements: Form and content

In case of requirements, we distinguish between

- their **external form** ("*syntax*"): How the requirement is described (language, text, image, formula, model, ...)
- their **content** ("*semantics*"): Which property it stipulates

Both are essential to achieve requirements of high quality.

We will discuss later what makes a requirement a good requirement.

More details in Lecture 2

Terms

- **Requirement artifact**: documented form of one (or more) requirements
- A **stakeholder** (in RE) is an individual/institution with interest in the project and/or the system to be created
 - e.g. user, administrator, security officer ...
 - and thus a potential **source of requirements**
- A **requirement source** designates the origin of a requirement, i.e. the real-world aspect, which causes the requirement (justification)
 - Stakeholders
 - (old) legacy systems in operation
 - Laws, regulations and standards, guidelines, ...
 - Literature
 - *More details are explained later in the course*
- A **requirement justification** (rationale) explains and justifies, why a certain requirement is demanded

Ideal view of RE: Iterative refinement and abstraction



How?

Ideally, **Top Down RE** is performed as follows:

Based on recorded and coordinated, generally formulated goals

- *The product must be internationally applicable*

rough requirements are established, compared and documented:

- *The product must be applicable in third world countries*
- *The product must be usable independently of the power supply*

then refined, concrete requirements are obtained

- *The system must be equipped with a dynamo that generates the necessary current; the voltage is ...*

and finally, a system specification is drawn up

- *A dynamo RT242pt of company D-Dyn is used..*



Why?

RE as part of product & system development

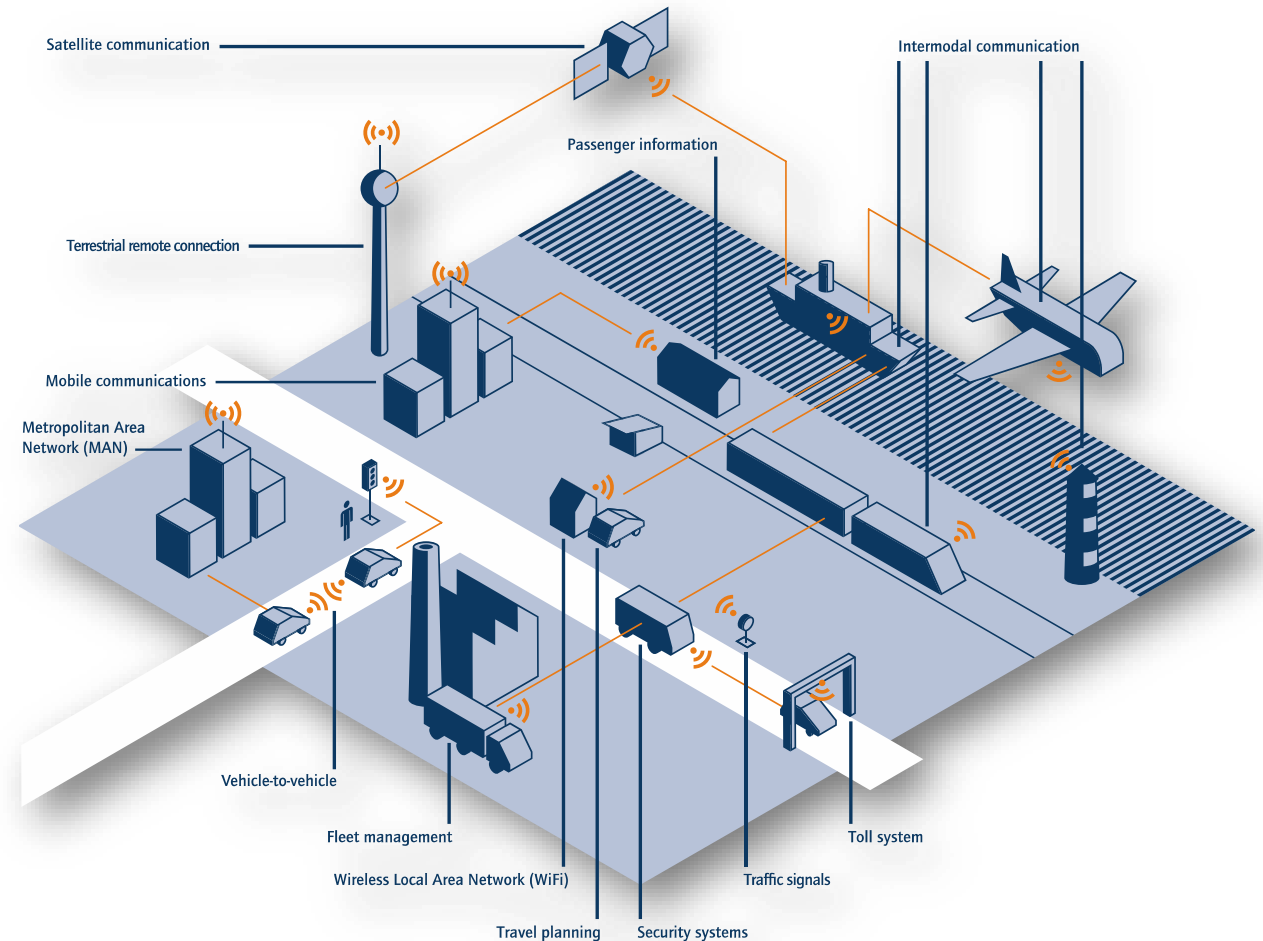
RE is not limited to software!

RE is generally aimed at

- Software systems
- Systems in general
(especially with embedded software)
- Products
- Processes

This lecture targets RE for:

- Software
- Software-intensive systems and products



„Health is not everything, but without health everything is nothing.“

- Arthur Schopenhauer, German philosopher, 1788-1860

Same holds for Requirements Engineering in Software Engineering!

RE results influence complete SE Process!

The **results of RE** are used in

- Designing the **architecture** of the system
- **Implementing** the functionality
- Quality assurance and **testing**
- Drafting **contracts** and acceptance tests
- Project organization and management
 - the **cost** estimate,
 - development **process**
 - **team** composition
 - the project planning (releases, test plan etc.)
 - **communication** with stakeholders (customer, users, development engineers, ...)

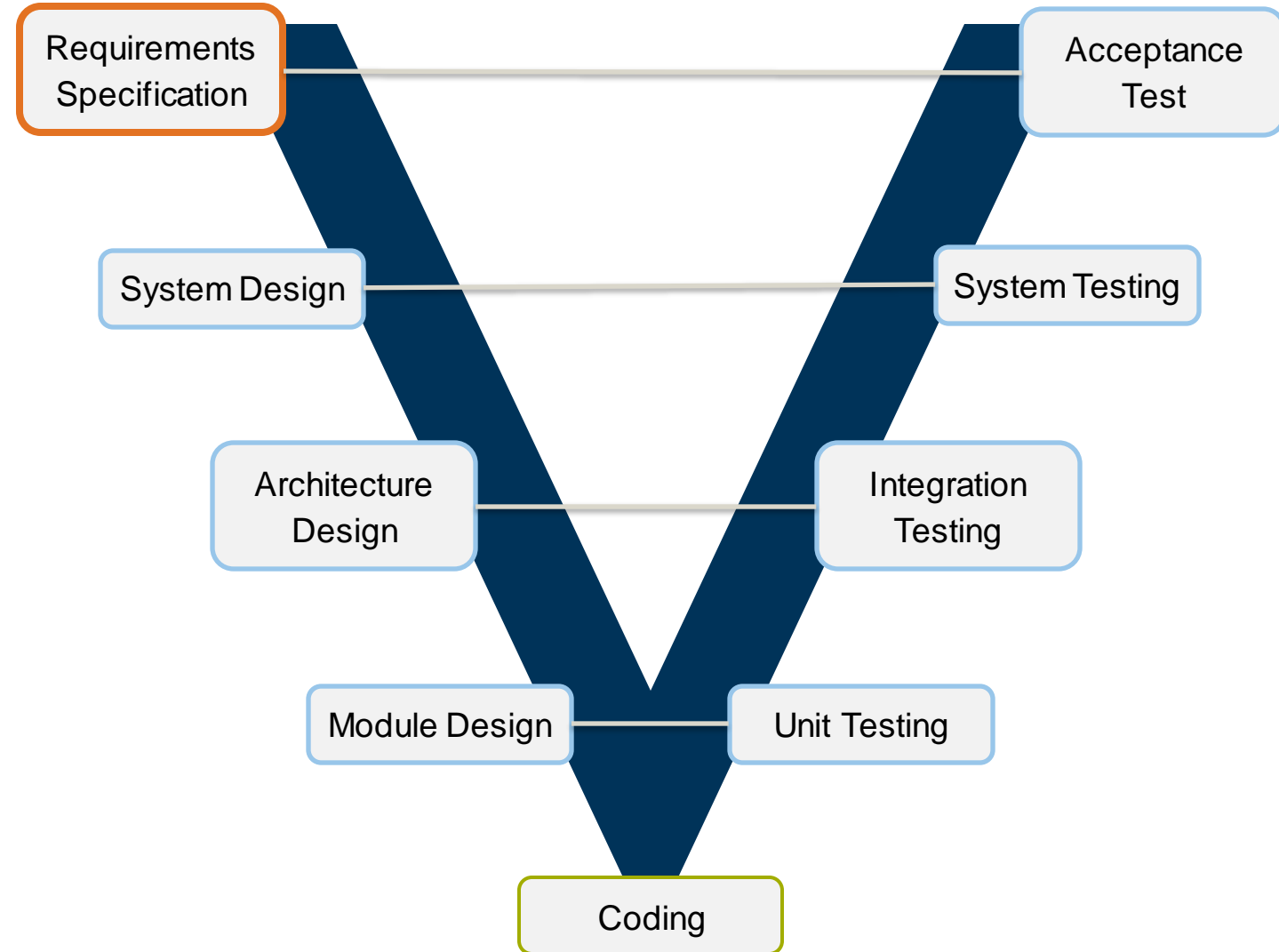
Error and correction costs

The **earlier an error** is **caused** and the later an error is **detected**, the more expensive is its removal.

Rule of thumb: factor 10 per step

Errors in Requirements Engineering are particularly **expensive**!

That's in part why agile approaches are so successful.



Success factor RE

Requirements as the linchpin for **project success**:

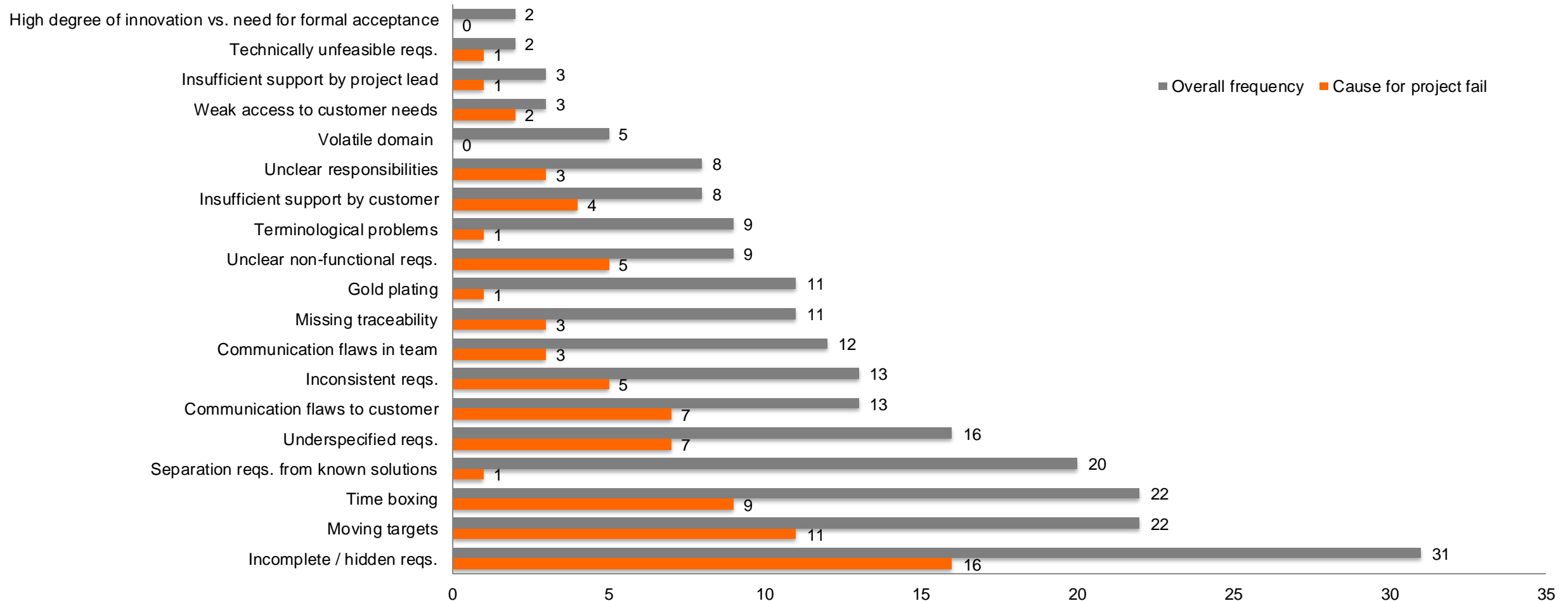
- If requirements are missing, the value of the product is reduced
- If unnecessary requirements are realized, the product becomes unnecessarily more expensive (Daimler, e.g., defined >1000 functions in the dev. process that were unnecessary from the customer's point of view)
- If requirements change during a project, costly changes and rescheduling are necessary

If the product is highly **innovative**, RE is particularly important:

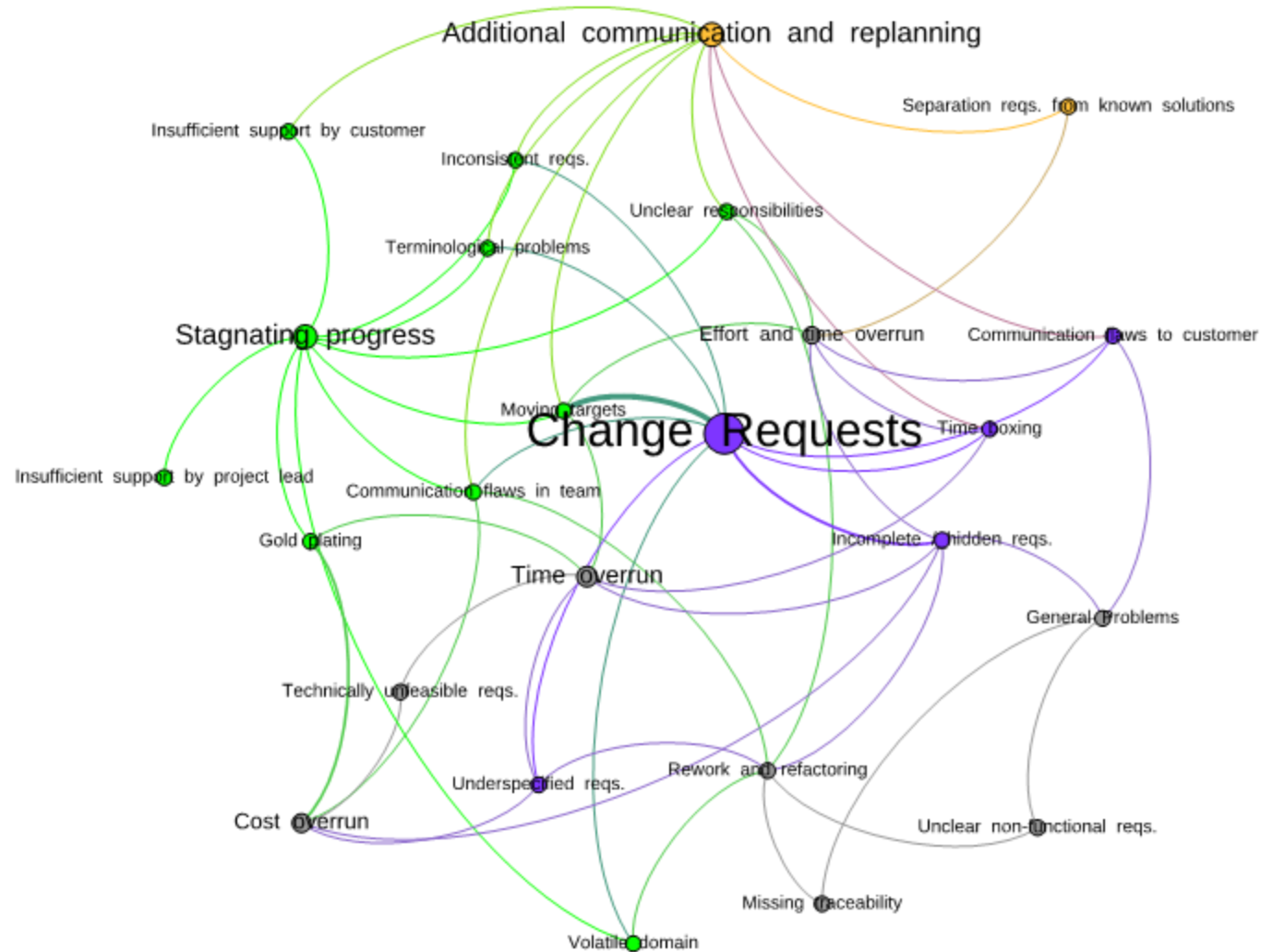
- Requirements are usually unclear at the beginning
- Requirements often unstable, as learning curves are to be expected

Problem in research: How to quantify exact **cost/benefit** ratio in RE?

Problems in requirements engineering as causes for project failure



There are also problems if the project did not fail entirely

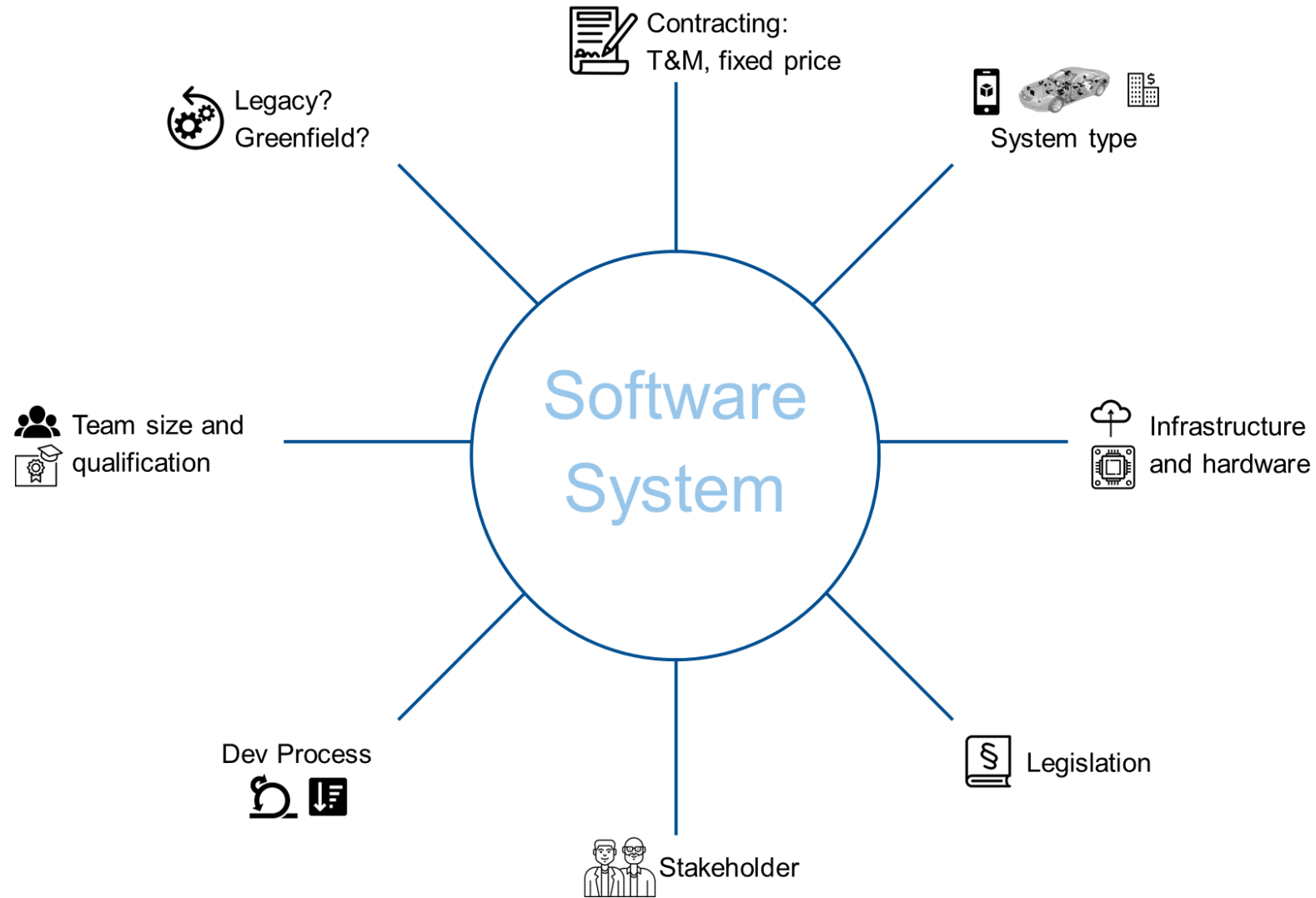


Discussion: Why is it so hard?

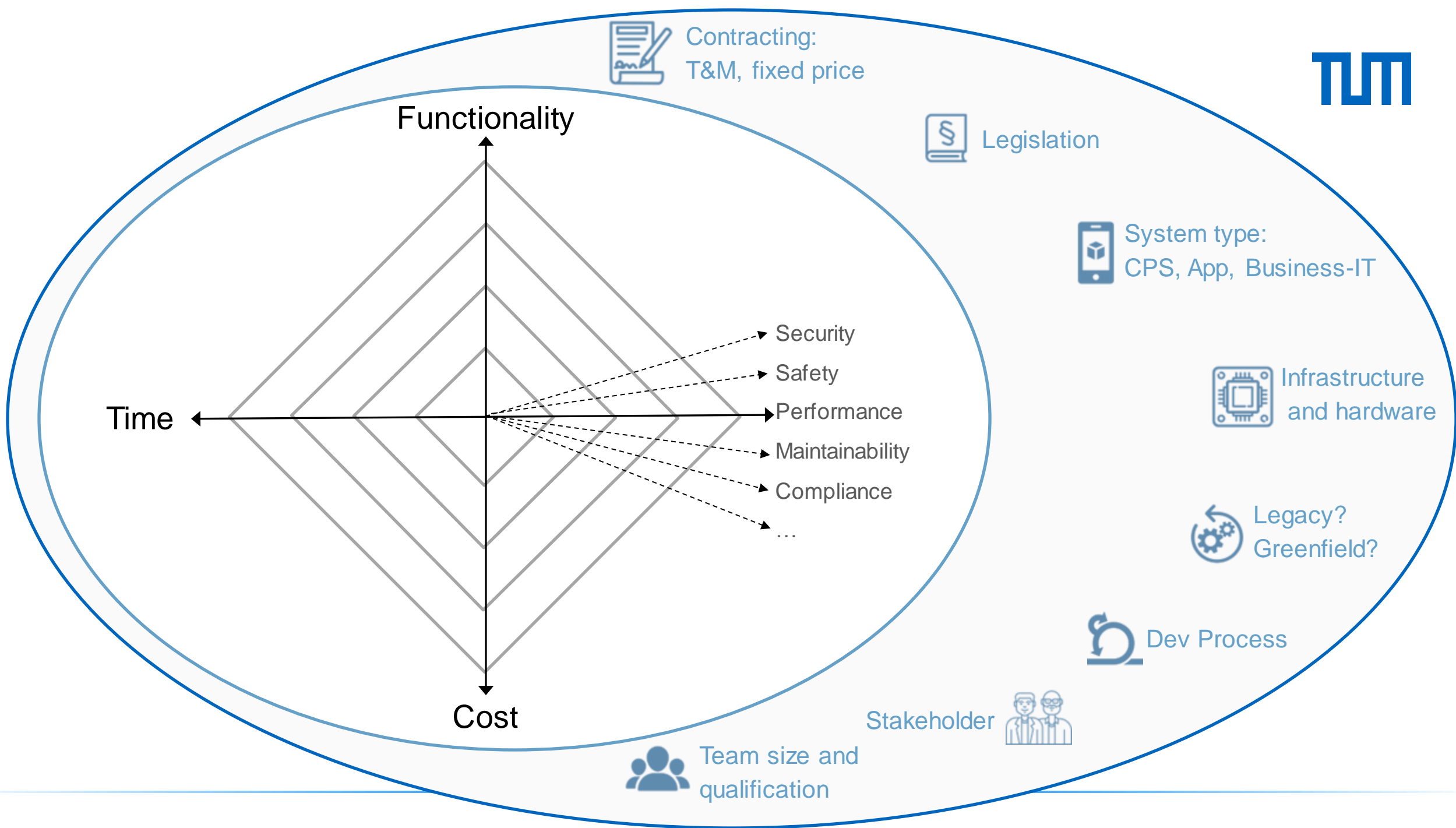


If Requirements Engineering is so important, why are there still so many failures?

Is there no best practice to do it?



There is not ,the one and only approach‘ in SE (or RE)



Limiting Context



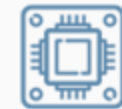
Contracting:
T&M, fixed price



Legislation



System type:
CPS, App, Business-IT



Infrastructure
and hardware



Legacy?
Greenfield?



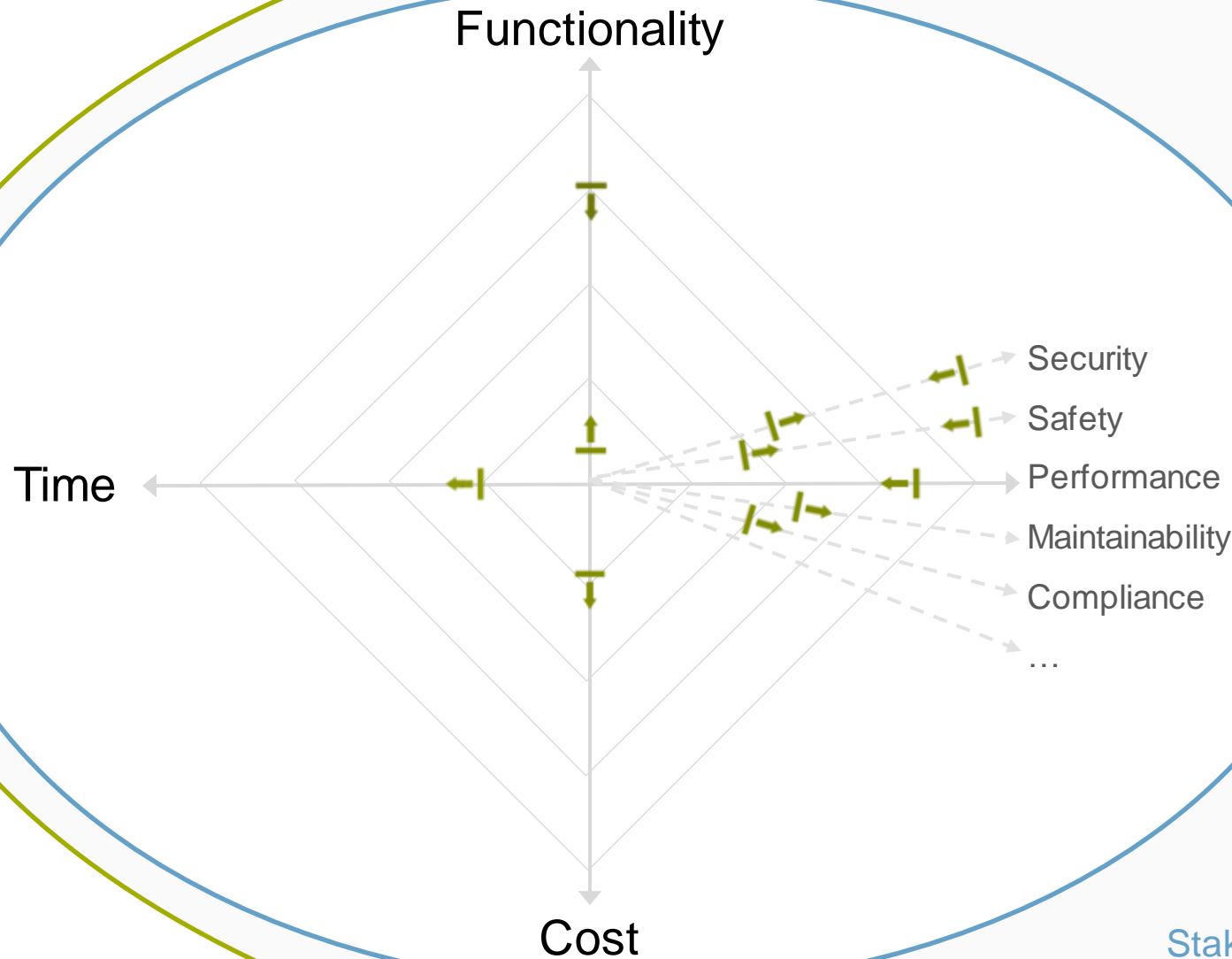
Dev Process



Stakeholder



Team size and
qualification



Limiting Context



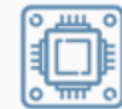
Contracting:
T&M, fixed price



Legislation



System type:
CPS, App, Business-IT



Infrastructure
and hardware



Legacy?
Greenfield?



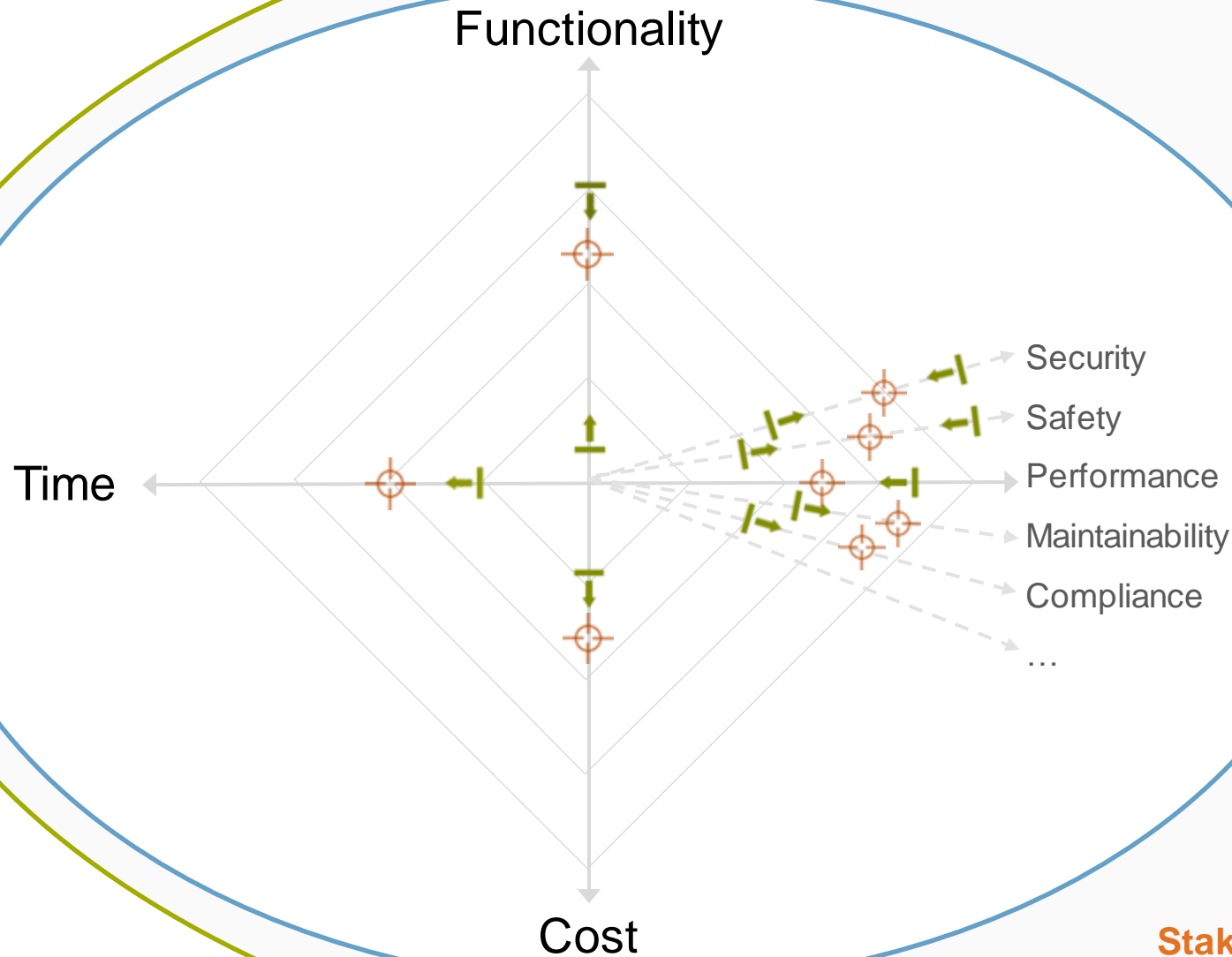
Dev Process



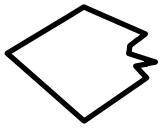
Stakeholder



Team size and
qualification



System



Contracting:
T&M, fixed price

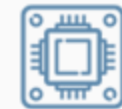
Limiting Context



Legislation



System type:
CPS, App, Business-IT



Infrastructure
and hardware



Legacy?
Greenfield?



Dev Process



Stakeholder



Team size and
qualification

Functionality

Time

Cost

Security

Safety

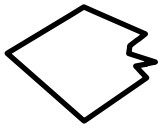
Performance

Maintainability

Compliance

...

System



Contracting:
T&M, fixed price

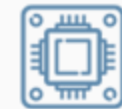
Limiting Context



Legislation



System type:
CPS, App, Business-IT



Infrastructure
and hardware



Legacy?
Greenfield?



Dev Process



Stakeholder



Team size and
qualification

Functionality

Time

Cost

Security

Safety

Performance

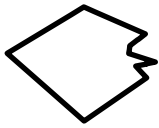
Maintainability

Compliance

...

! Change ➡

System



Contracting:
T&M, fixed price

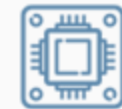
Limiting Context



Legislation



System type:
CPS, App, Business-IT



Infrastructure
and hardware



Legacy?
Greenfield?



Dev Process



Stakeholder



Team size and
qualification

Functionality

Time

Cost

Security

Safety

Performance

Maintainability

Compliance

...



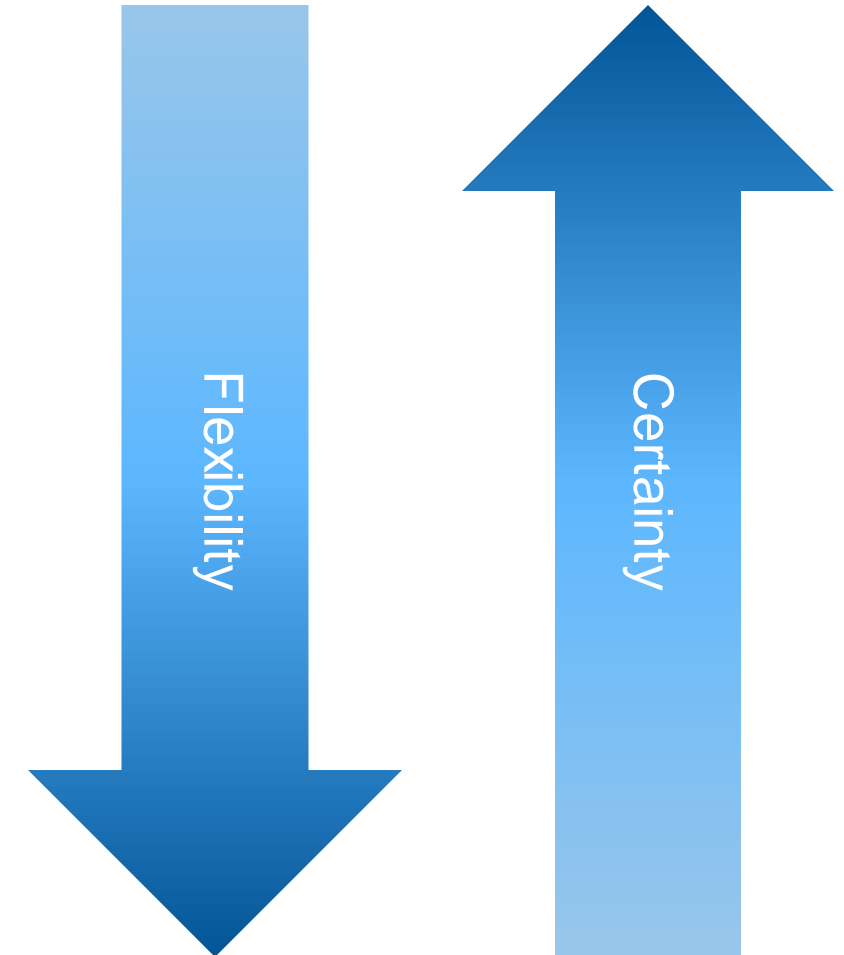
Change

Different starting points for Requirements Engineering

Interface Engineering

Re-Engineering

Greenfield Engineering



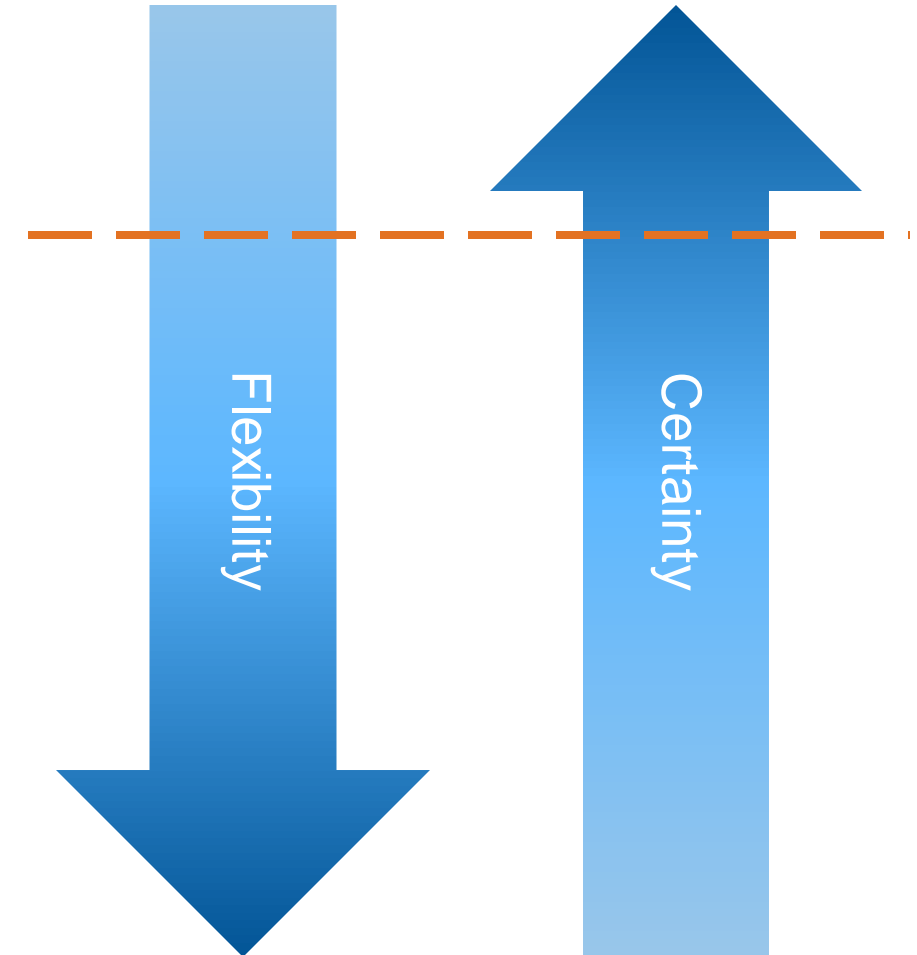
Different starting points for Requirements Engineering

Interface Engineering:

New functionality is added to an existing and functioning system.

Usually, adapting to new technologies or new business models.

Example: adding an interface for the next generation of smartphones

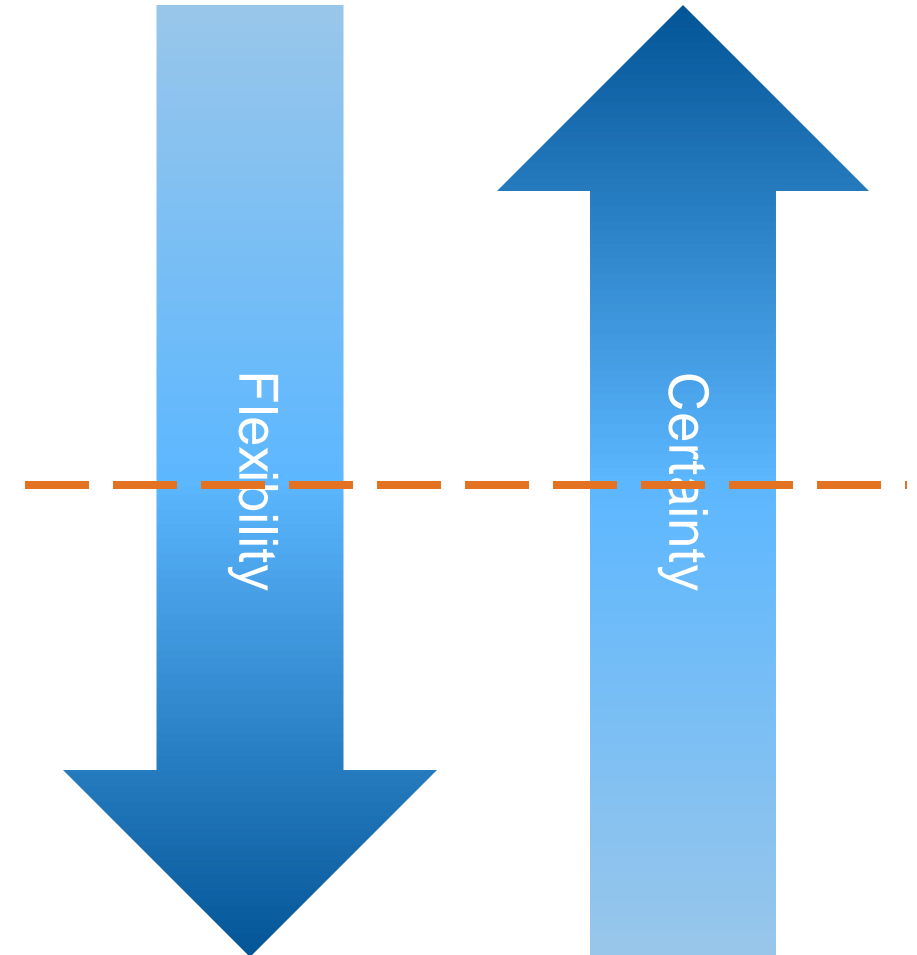


Different starting points for Requirements Engineering

Re-Engineering:

An existing system is entirely replaced by a new one.
Usually, this is triggered by high maintenance costs for the old system

Example: Re-implement a Cobol mainframe system as a cloud-based Java application

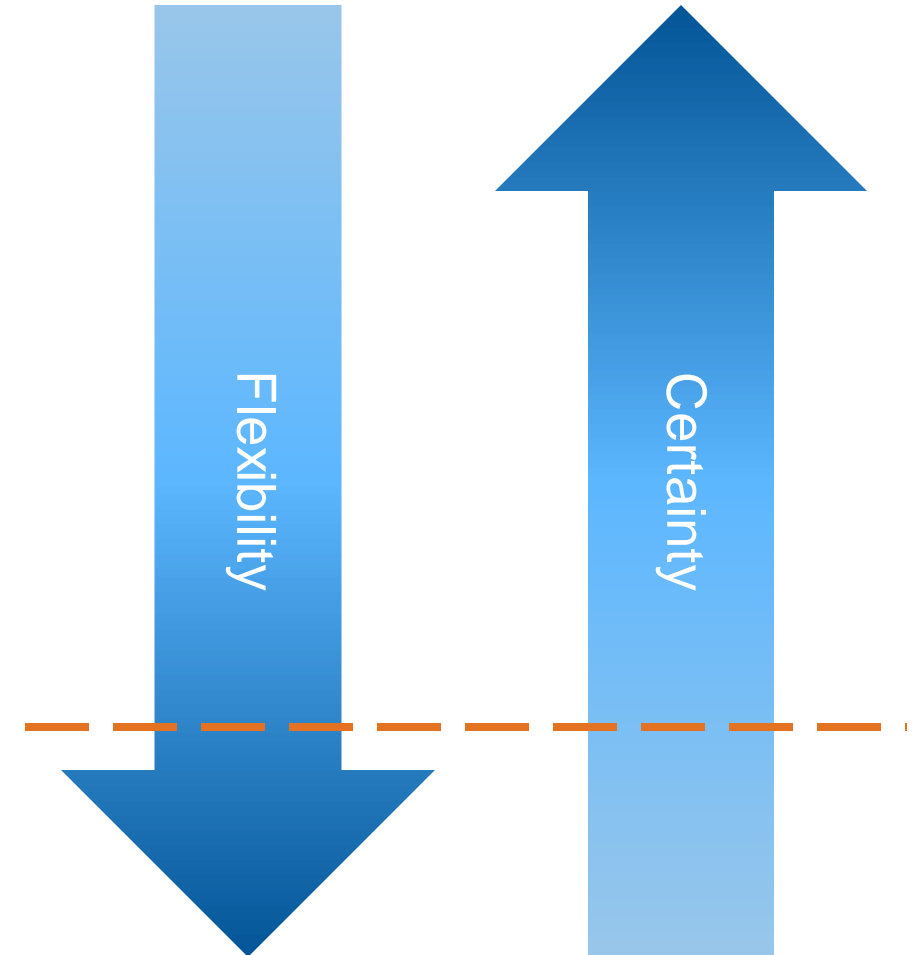


Different starting points for Requirements Engineering

Greenfield Engineering:

There is no existing system for reference. Everything is developed from scratch.

Example: Digitization of previously manual processes

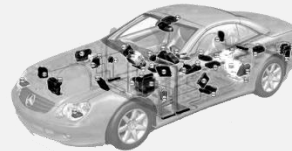


Different system types and application domains

Embedded Software



Cyber Physical System



Business IT

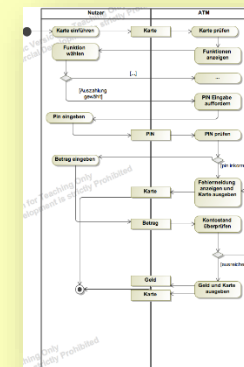


Change
-driven



vs

Process
-oriented



Discussion: Different System Types



*What **kind of system** are we building in our „Rent-a-Scooter“ project?*

Reasons for Change

Market demands, requirements and context change:

This led to the agile movement of software development ("Embrace Change").

Competition:

Competitors are fast and aggressive. Late in the market, you lose.

Configurability:

Software-defined-X makes hardware more flexible: configuration *allows* for change.

RE in an agile world

What we have seen so far is mostly aligned to waterfall or V-model

1968: The first Software Engineering Conference established:

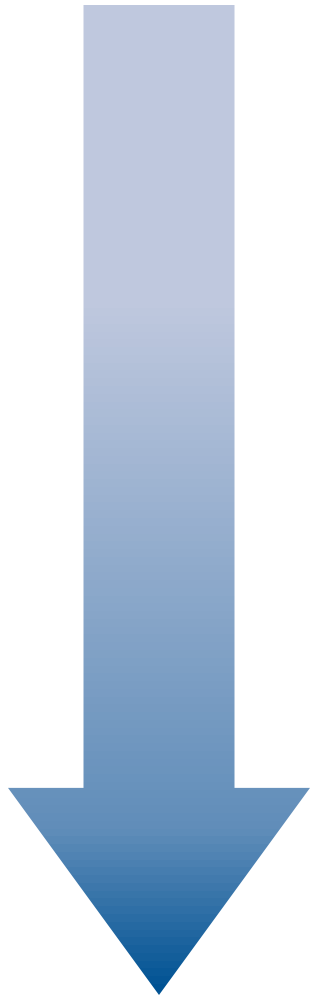
- Industrialization of software development
- **Separation of planning and production**

Today: This separation is weakened in the context of agility: Iteration with short cycles and concentration on immediate customer benefit

- Always executable code and **continuous delivery**
- „40% of the requirements become **obsolete within 1 year!**“

Does RE still work the same?

Outline and Outlook



Terms and Definitions

Reasons why Requirements Engineering is important

Context-specific nature of SE and RE

Quality models for requirements

Engineering models in RE

Stakeholder and Requirements Elicitation

Goals and Goal-oriented RE

Non-functional requirements

Functional requirements

Formalization

Agile Processes

Requirements Management and Quality Assurance

Trends in Research