fortiss

Ihr Ansprechpartner

Dr. Jannik Fischbach

Postdoctoral Researcher

fortiss GmbH

Guerickestr. 25

80805 München

fischbach@fortiss.org

Agenda

1. Kritikalität und Grundlagen des Requirements Engineering
2. Modellierung von Anforderungen
3. Einführung in effektives Prompt Engineering
4. Offene Fragerunde

Competence Field Requirements Engineering

Our typical activities for requirements engineering approaches and tools

Research

Problem-oriented, empirically informed research and benchmarks

Disclosing open data sets on evidence-based research

Application

Artefact-based RE approaches as engineering guidelines

Tool support for automated compliance checks

Impact

Knowledge transfer workshops and conferences on selected domains and topics

Application of different education and training formats (for professionals and in Higher Education)

Relevance of Requirements Engineering

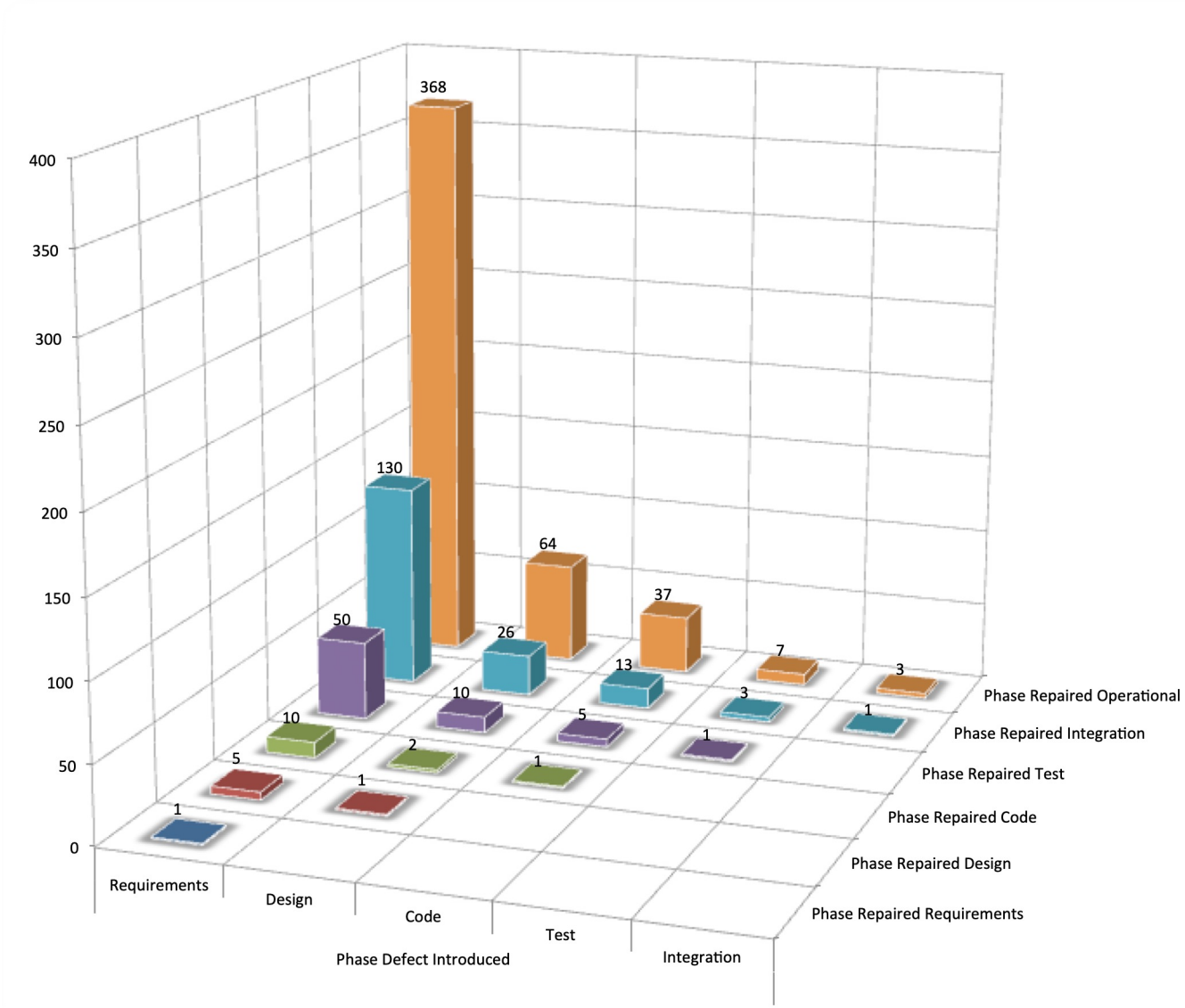
For a software system, RE (and the outcomes of the RE process) influences

- The system's functionality and quality
- The system's cost
- The system's usefulness
- The system's complexity

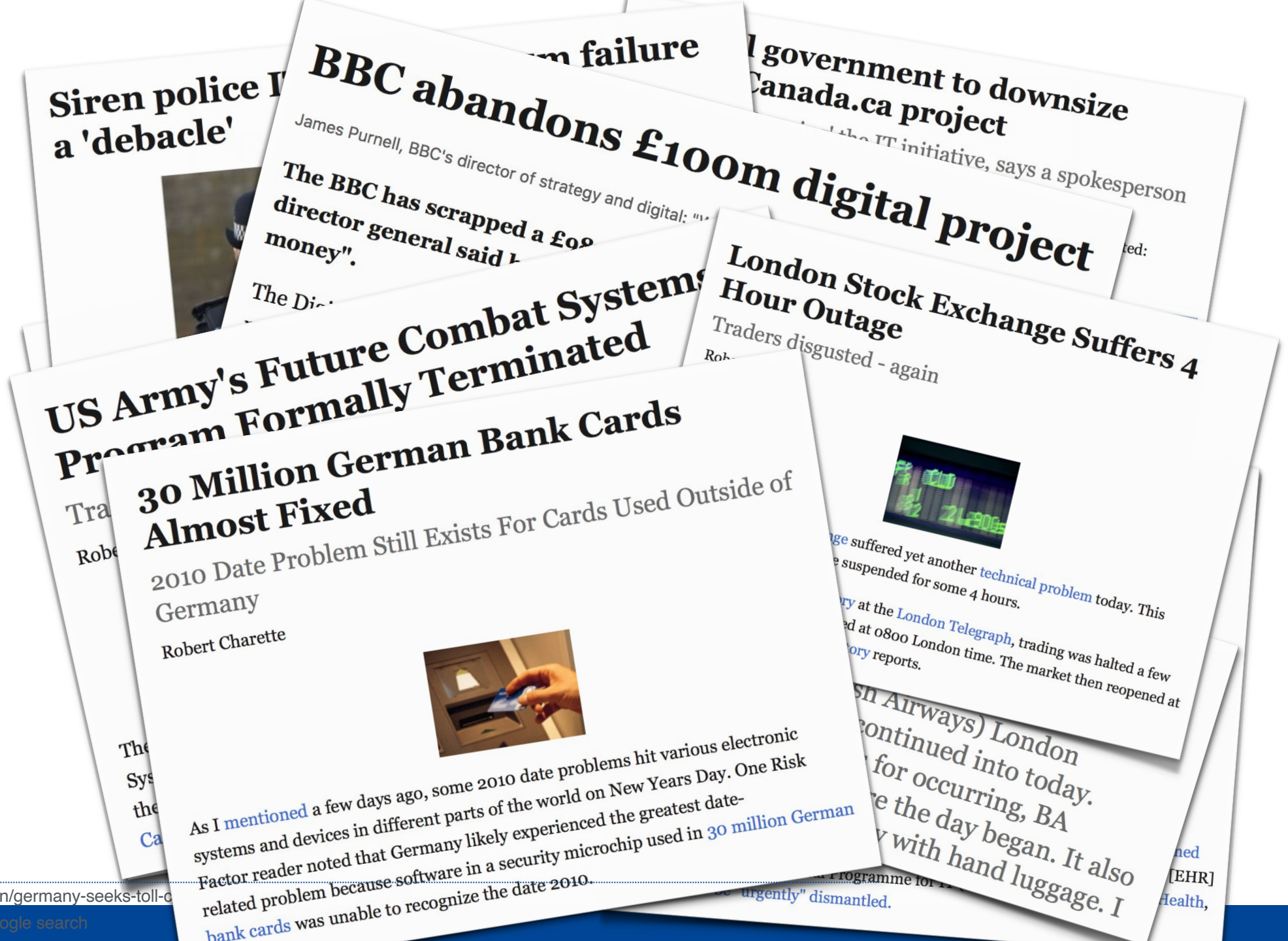
But: RE is often underestimated and not taken seriously. Often, engineers are not even in sufficient command of the RE process…

RE as a success factor

- The earlier an error is introduced (e.g., in RE) and the later the error is detected, the more expensive it is to fix it
- **Rule of thumb:**
  factor 10 per level



Source: Bennet/Wennberg (NASA project analysis)



[1] http://www.dw.com/en/germany-seeks-toll-c

[2] Quick and simple Google search

33%

… of errors happen in RE.

36%

… of errors in RE lead to project failure.

An initial categorisation of requirements

Functional Requirements

- Behaviour of a system (general)
- Behaviour from the perspective of the user

Quality requirements

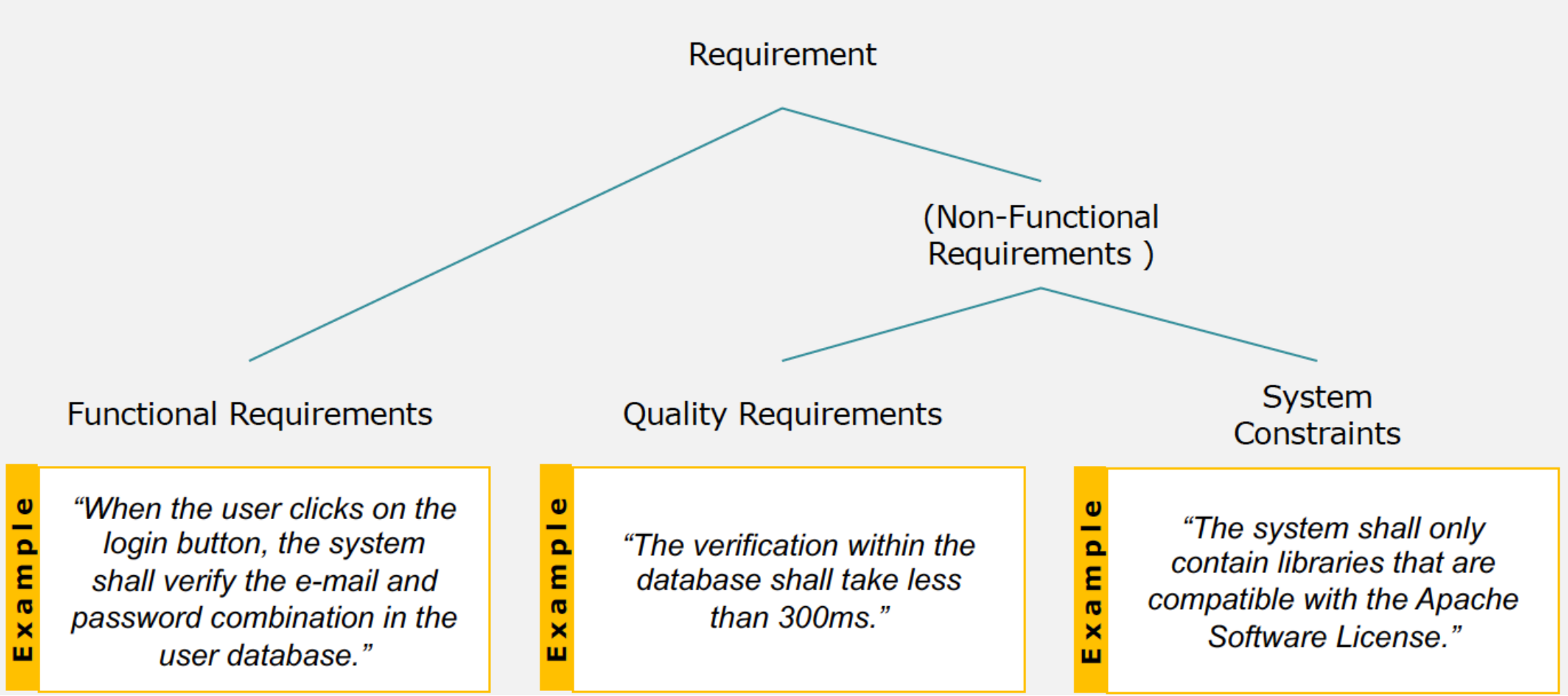
- Quality characteristics and properties of a system
- (Ideally) quantified properties regarding the behaviour

Often called "Non-functional requirements"

Process requirements (project-specific)

- Requirements towards the development process (scheduling, milestones, budget, etc.)
- Definitions/rules for the implementation (constraints, components to (re-)use etc.)

An initial categorisation of requirements



Definition: Requirements Engineering (RE)

> **Requirements Engineering (RE)** is the systematic, iterative, and disciplined approach to develop an explicit requirements and system specification that all stakeholders agree upon.

Elementary tasks include, but are not limited to:

- Requirements elicitation (identifying all relevant requirements)
- Requirements analysis (understanding requirements and achieving consensus)
- Requirements specification (structuring, modelling, documenting requirements)
- Requirements validation and verification (ensuring validity and quality)

Basis: M. Glinz. A Glossary of Requirements Engineering Terminology, v1.6, IREB, 2014

RE typically concerned with “problem space”

Requirements engineering aims at the **problem space**…

- at framing the problem for a development task as comprehensively and as precisely as possible: *Why is something necessary?*
- To derive appropriate requirements towards the desired solution and agreement between stakeholders: *What (capabilities, properties) is necessary?*

Design and implementation, in contrast, aims at the **solution space**,
i.e. development and evolution of of possible solutions of: *How will the solution* be realised?*

Differentiating the problem space from the solution space is essential!

Definition: Requirements Management (RM)

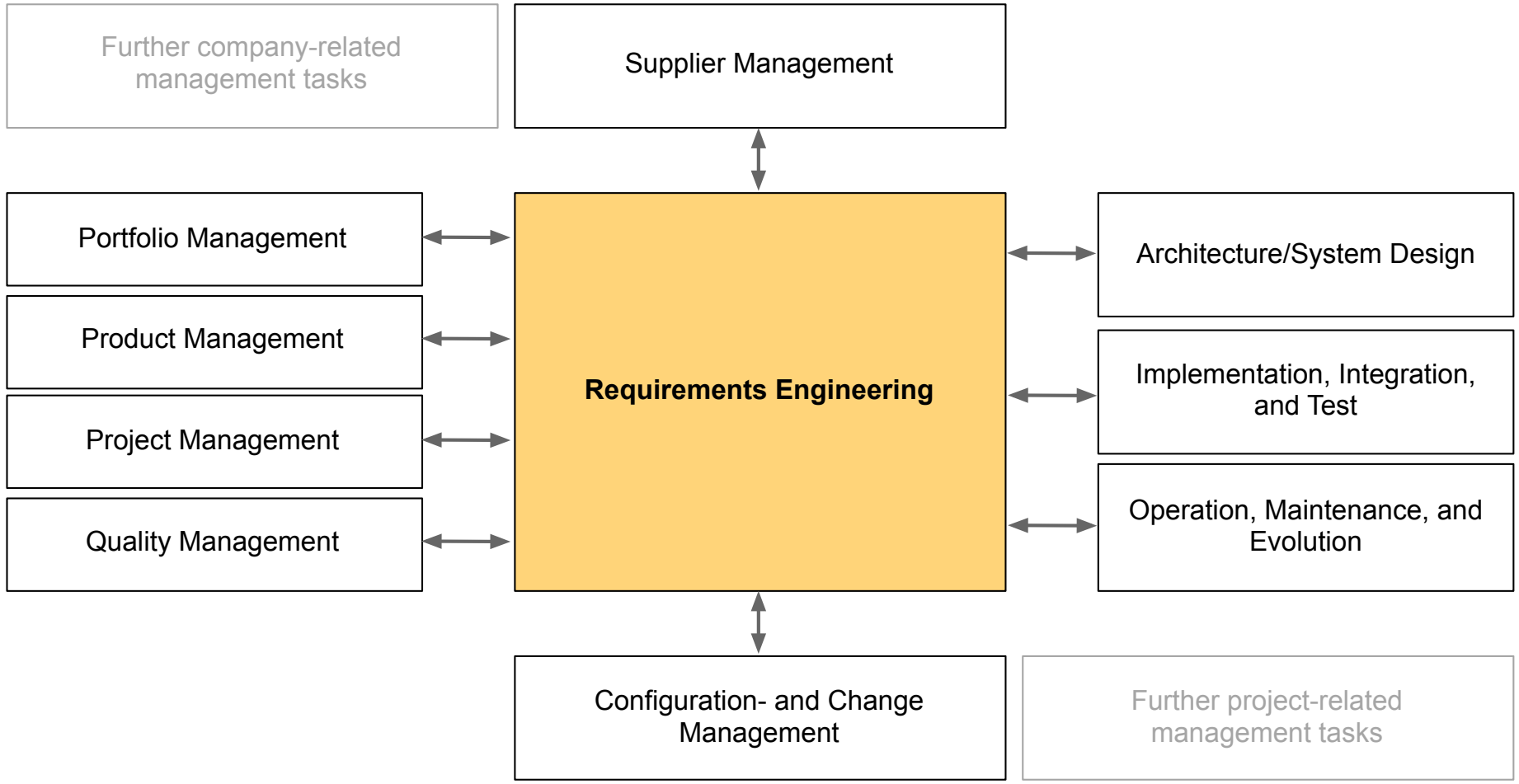
Requirements Management (RM) aims at efficiently and effectively managing and using requirements along the whole system lifecycle.

Elementary tasks include, but are not limited to:

- Archiving and baselining requirements
- Modifying requirements due to new knowledge/insights
- Tracing and verifying requirements (impact analysis, support of change processes)

Basis: M. Glinz. A Glossary of Requirements Engineering Terminology, v1.6, IREB, 2014

RE and RM build a key interface to several activities in the development life cycle



Agenda

1. Kritikalität und Grundlagen des Requirements Engineering
2. Modellierung von Anforderungen
3. Einführung in effektives Prompt Engineering
4. Offene Fragerunde

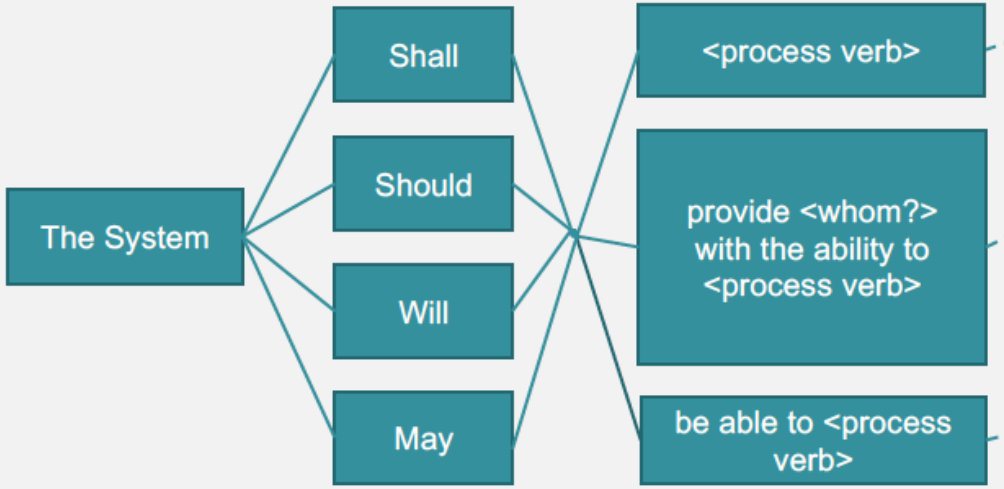
Requirement Documentation Techniques

Technique	Advantages	Disadvantages
Natural language	• Able to express any kind of requirement perspective • No training required	• Increase the likelihood of ambiguous requirements • Increase the likelihood of mixing up requirements types and perspectives • Sometimes hard to keep an overview
Models	• Easy to understand by a trained user • Lowers the degree of ambiguity • Offers different levels of abstraction based on intended purpose	• Require training. • One model cannot be used to express all requirements perspectives • More difficult to trace
Combination of natural language and models	• Disadvantages of the other exploits advantages of both. E.g. Models are complemented by the use of natural language and vise versa.	• Increase the likelihood of mixing up requirements types and perspectives • Risk of inconsistencies

68

Requirement Documentation Techniques

- System Activity Types:
  - Autonomous System Activity
  - User Interaction
  - Interface
- Inform the requirement template type



Example: The RSP system SHALL validate

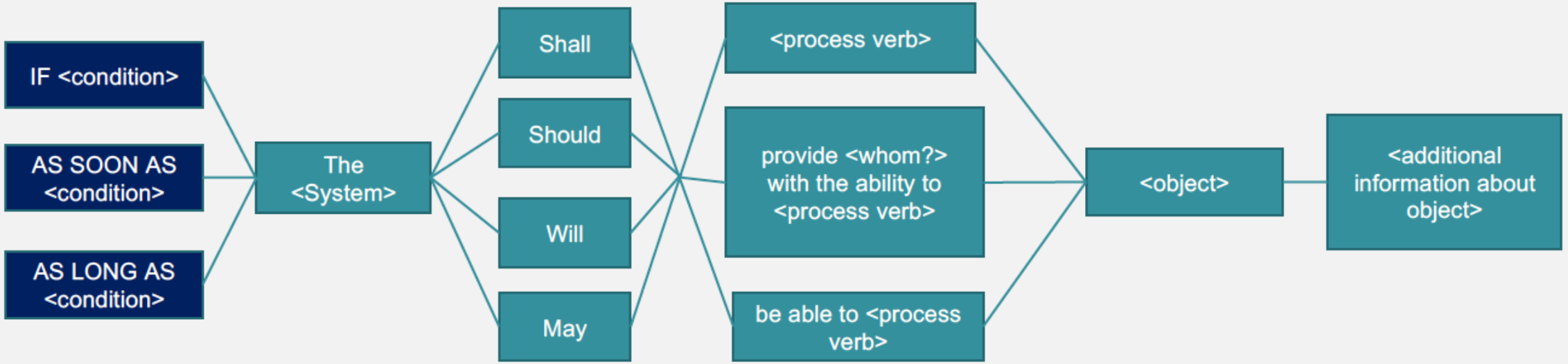
Example: The RSP system shall provide a user with the ability to search.

Example: The RSP system will be able to parse

Requirement Documentation Techniques

Constraints under which processes take place

- Logical conditions use logical conjunction ‘if’
- Temporal conditions use Temporal conjunction ‘As soon as’ (Event) or ‘As long as’ (Time Period)
- Why NOT ‘When’ ? (-> to prevent ambiguity)



Model-Based Requirements Documentation

Technique	Advantages	Disadvantages
Natural language	• Able to express any kind of requirement perspective • No training required	• Increase the likelihood of ambiguous requirements • Increase the likelihood of mixing up requirements types and perspectives • Sometimes hard to keep an overview
Models	• Easy to understand by a trained user • Lowers the degree of ambiguity • Offers different levels of abstraction based on intended purpose	• Require training. • One model cannot be used to express all requirements perspectives • More difficult to trace
Combination of natural language and models	• Disadvantages of the other exploits advantages of both. E.g. Models are complemented by the use of natural language and vise versa.	• Increase the likelihood of mixing up requirements types and perspectives • Risk of inconsistencies

107

Model-Based Requirements Documentation

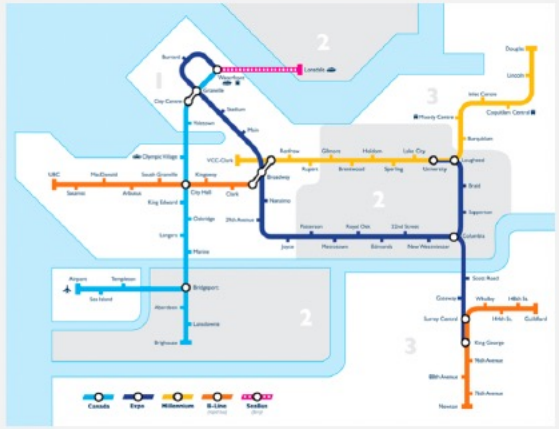
Try to describe this map in natural language ☺

The reduction in information is rather high (no distance or actual location) But still we can extract the information we want.

The language is simple:
- Colored lines repetend the metro track
- A station is a name
-The end stations are in bold

Model-Based Requirements Documentation

Munich (Germany)

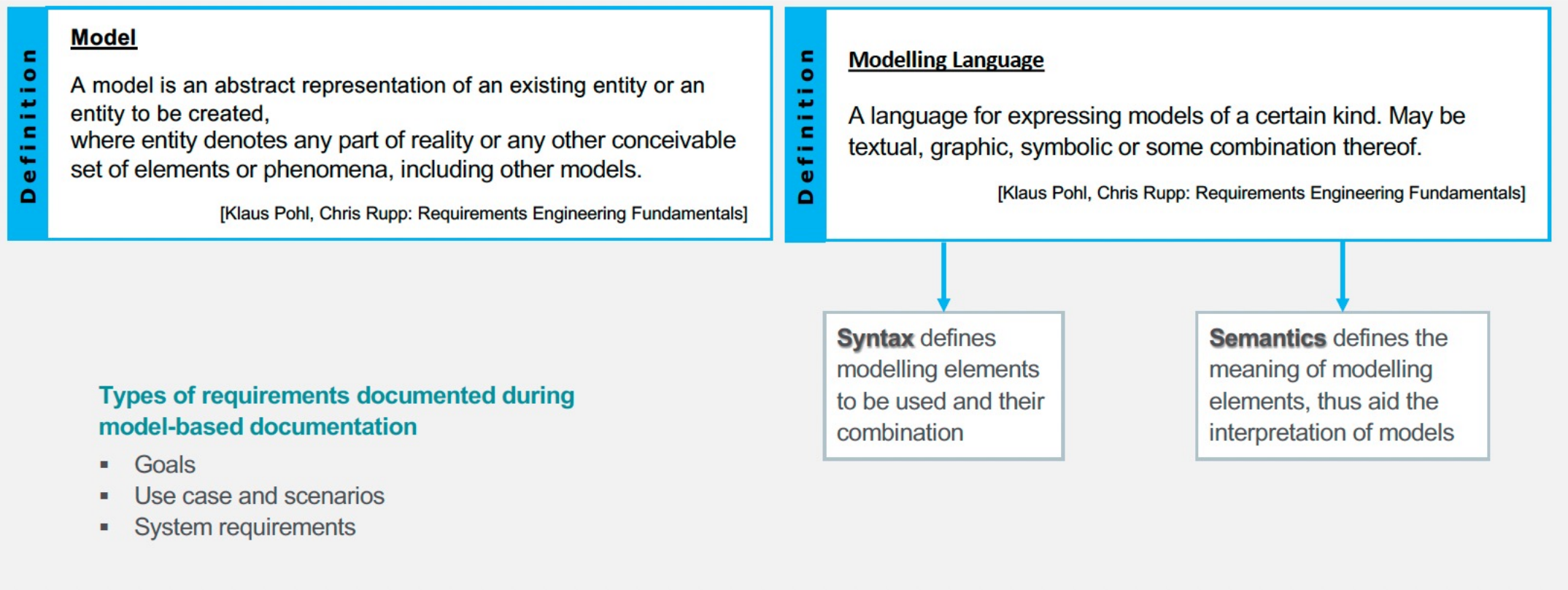


Vancouver (Canada)

The language is called “Transit Map”
Everywhere in the World you can extract the same information

New York (USA)

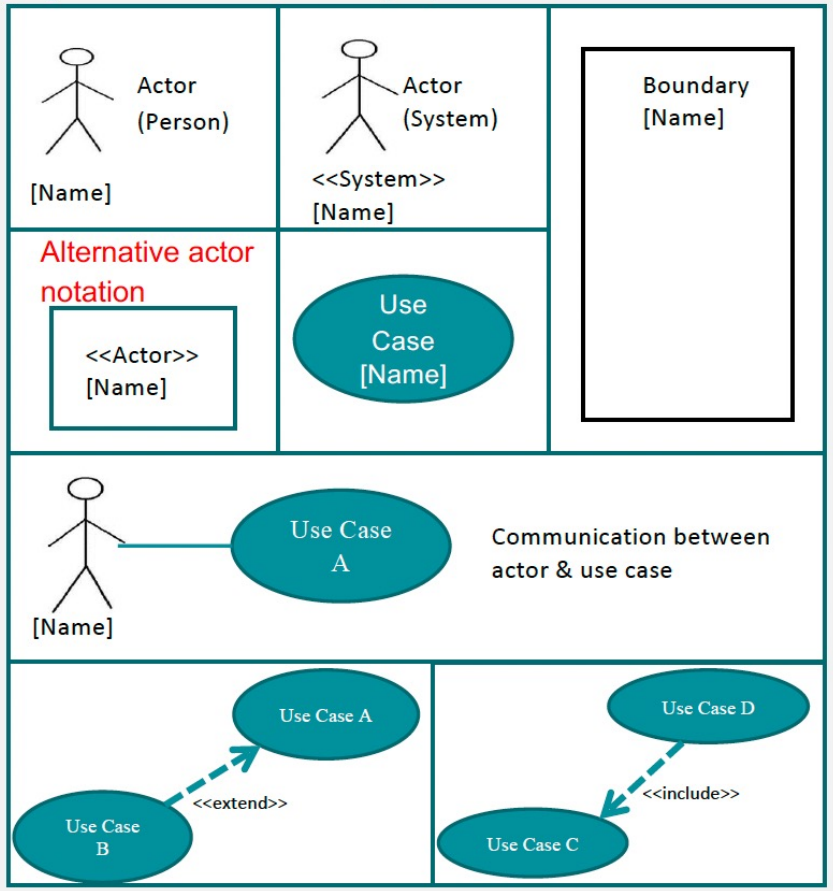
Model-Based Requirements Documentation



Model-Based Requirements Documentation

Use Case Overview Diagram Modeling Elements:

Objects and relationships:



Extend, Include and extension point

- **Extend** -> Adds *optional* behavior to the use case, the extended use case can exists without the extending use case.

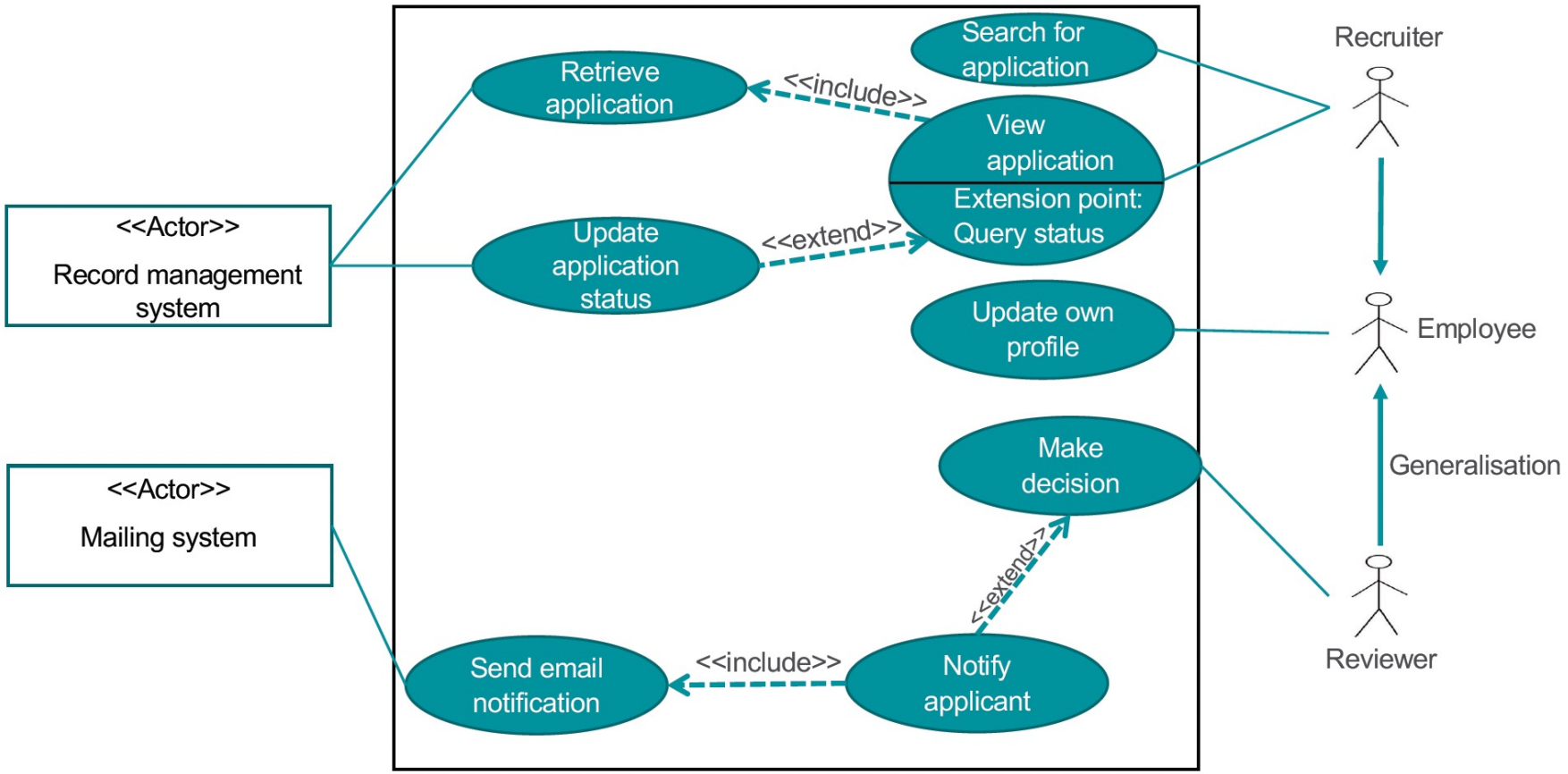
  This optional behavior is knowns as an **extension point** within the extended use case.
- **Include** -> Adds necessary behavior to the use case, use case cannot exist without the included behavior.

Example:

- Extends -> Get Help, Report application for illegal activity
- Include -> Enter data, verify user, …

118

Model-Based Requirements Documentation



Agenda

1. Kritikalität und Grundlagen des Requirements Engineering
2. Modellierung von Anforderungen
3. Einführung in effektives Prompt Engineering
4. Offene Fragerunde

“Prompt Engineering is the art of communicating with a generative large language model.”

Prompt Engineering – Grundlagen (1/2)

Was sind LLMs?

- Transformer-basierte neuronale Netzwerke
- Trainiert auf enormen Textmengen
- Kontext-Verständnis und kohärente Antworten

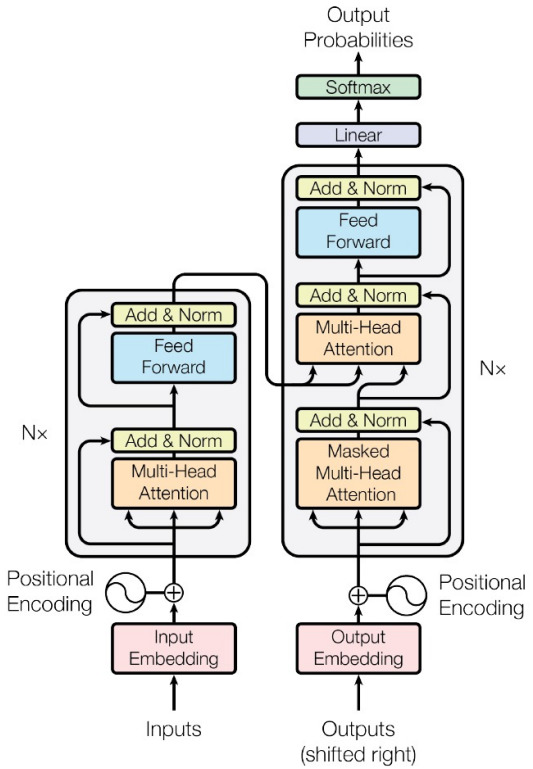
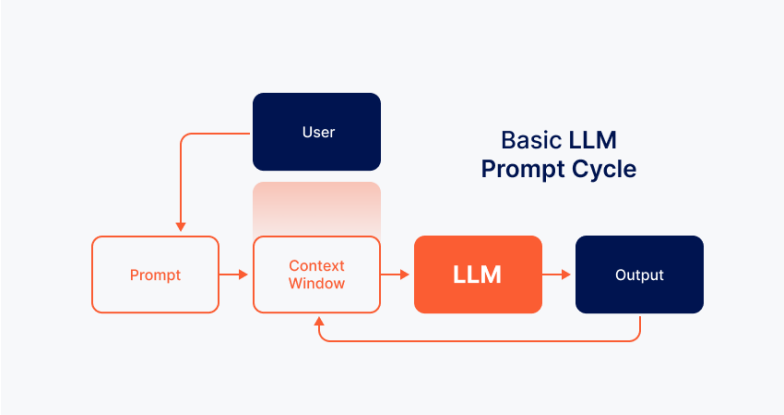


Figure 1: The Transformer - model architecture.

Warum Prompts?

- Brücke zwischen menschlicher Intention und KI-Verständnis
- Textuelle Eingabe zur Kommunikation mit LLMs
- Bestimmt Qualität und Relevanz der Ausgabe



Prompt Engineering – Grundlagen (2/2)

Was ist Prompt-Engineering?

= die Disziplin der Gestaltung und Optimierung von Eingabeaufforderungen (Prompts),
um Sprachmodelle (LLMs) effizient für eine Vielzahl von Aufgaben zu nutzen.

Ziel: Verbesserung der Modellleistung bei Aufgaben wie Fragebeantwortung, Textgenerierung und Codegenerierung.

Warum ist es wichtig?

- Verbessert die Genauigkeit und Relevanz der Outputs
- Ermöglicht die Steuerung des Modellverhaltens
- Erschließt das volle Potenzial von LLMs

Grundlagen des Prompts

Zwei Hauptansätze

Zero-Shot Prompting

Das Modell erhält eine Anweisung **ohne jegliche Beispiele.**

Beispiel-Prompt:
Übersetze den folgenden Satz ins Englische: „Ich liebe Prompt Engineering.“

-> Geeignet für einfache und direkte Aufgaben

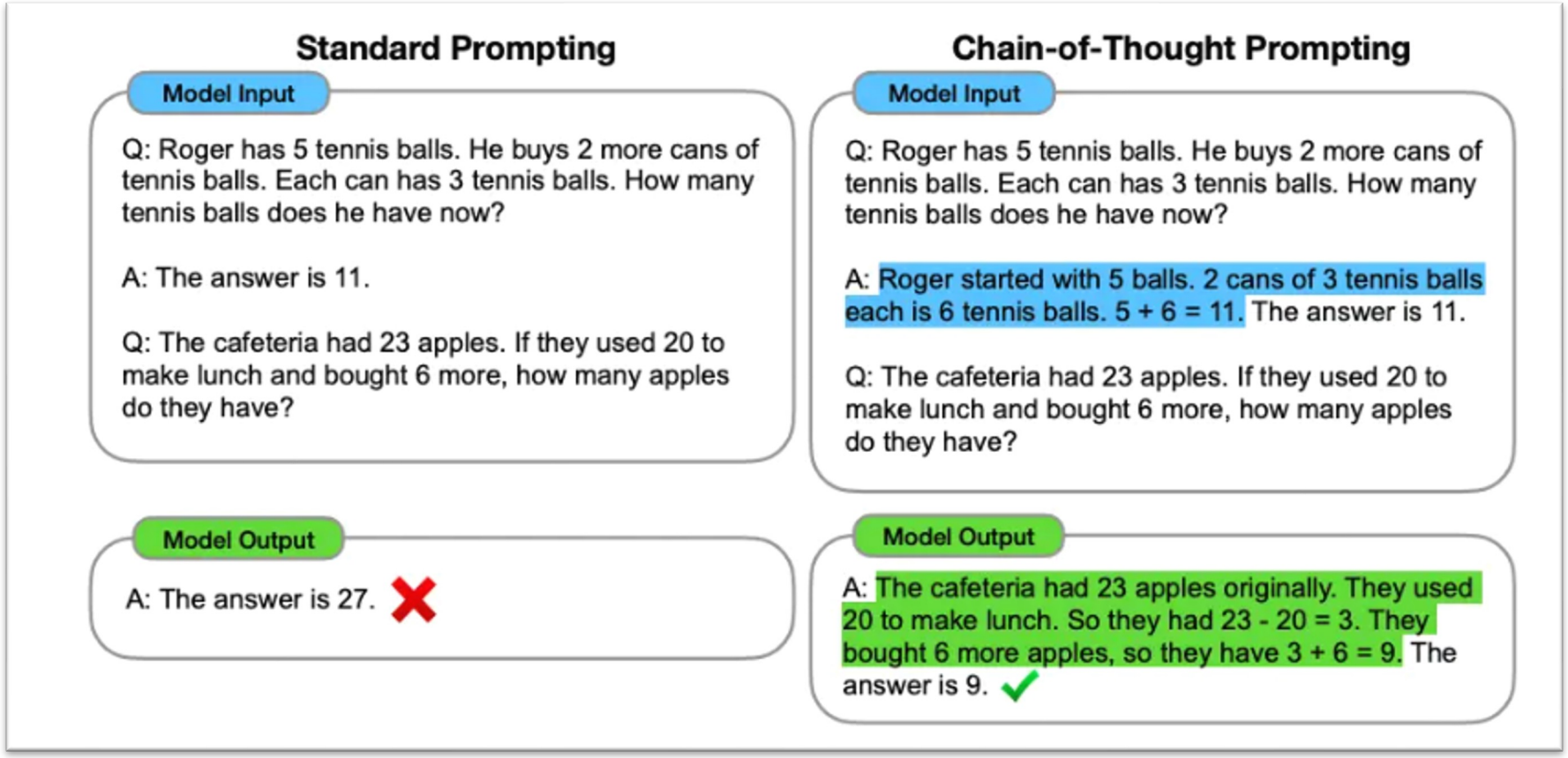
Few-Shot Prompting

Dem Modell werden einige Beispiele (Shots) zur Verfügung gestellt, um den Kontext und das gewünschte Ausgabeformat zu demonstrieren.

Beispiel-Prompt:
"Deutsch: 'Hallo', Englisch: 'Hello‘.
Deutsch: 'Wie geht es Ihnen?', Englisch: 'How are you?‘.
Deutsch: 'Was ist ein Apfel?', Englisch: „

-> Verbessert die Leistung bei komplexeren oder spezifischeren Aufgaben

Chain-of-Thought Prompting



Retrieval-Augmented Generation (RAG)

Externe Daten als Schlüssel zum Erfolg

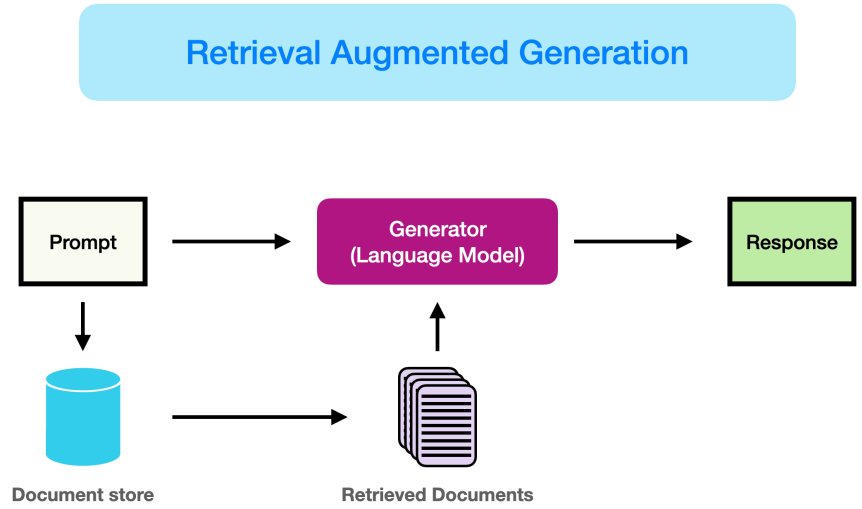
Wissenslücken von LLMs:

- Zeitliche Begrenzung
- Unternehmensspezifisch
- Personalisiert
- Dynamisch

Wo RAG unverzichtbar ist:

- "Aktuellen Projektstatus abrufen"
- "Letzte Quartalszahlen analysieren"
- "Kundenspezifische Lösungen vorschlagen"
- "Nach neuesten Gesetzen bewerten"

Retrieve → Augment → Generate



Warum RAG notwendig ist

- LLMs haben einen **festen Knowledge Cutoff** und können nicht auf aktuelle oder spezifische Informationen zugreifen
- RAG ermöglicht es, die Fähigkeiten von LLMs mit externen, aktuellen Datenquellen zu erweitern

Komponenten eines effektiven Prompts

Komponenten für maximale Modellleistung

Anweisung: klare Aufgabe oder Auftrag.

Kontext: Hintergrundinformationen zur Steuerung der Antwort.

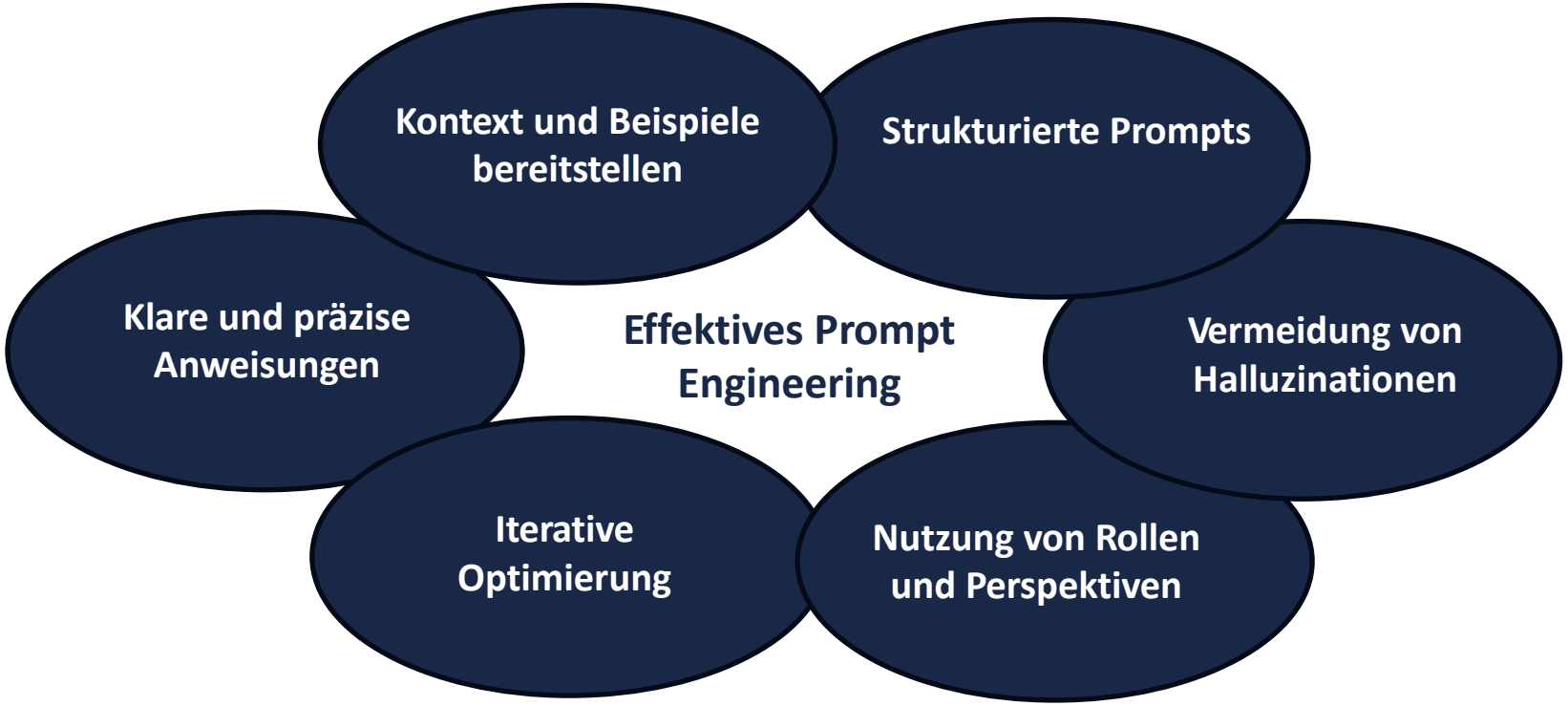
Eingabedaten: das konkrete Objekt oder die Frage.

Ausgabeindikator: Format oder Stil der Antwort.

Generelle Tipps:

- **Start Simple**: Iterative Verbesserung durch Experimentieren
- **Klare Instruktionen**: Präzise Befehle mit strukturierter Formatierung
- **Spezifisch sein**: Je detaillierter, desto besser die Ergebnisse
- **Ungenauigkeiten vermeiden**: Direkte und konkrete Kommunikation
- **Positiv formulieren**: Sagen was zu tun ist, nicht was zu vermeiden ist

Optimierung der Interaktion mit KI-Modellen durch effektives Prompting

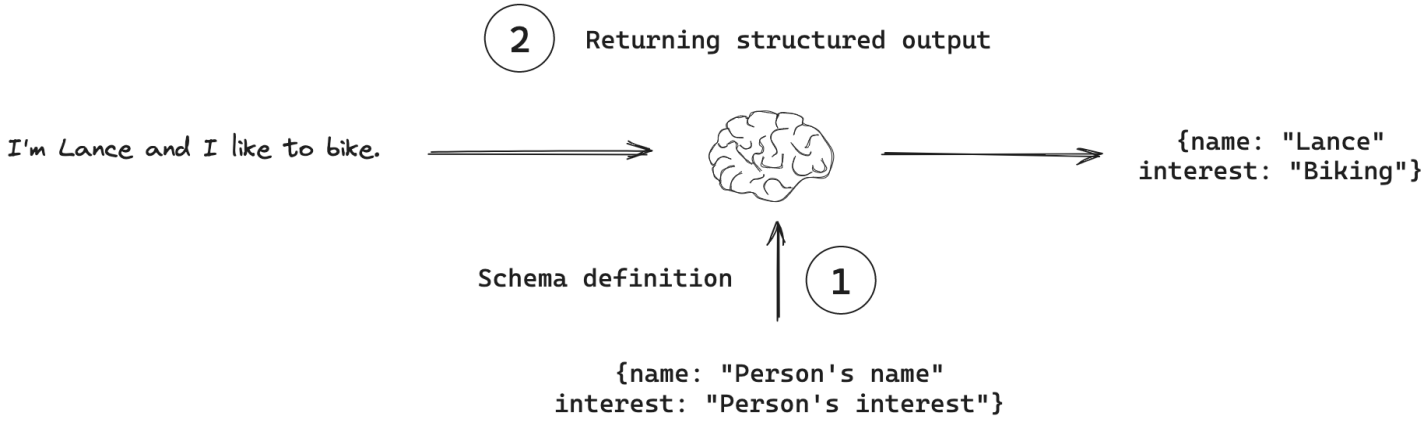


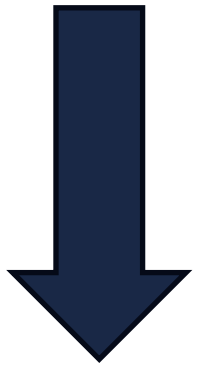
Strukturierte Outputs – Warum sie wichtig sind:

Problem mit unstrukturiertem Text:

- Unzuverlässigkeit
- Schwierige Validierung
- Hoher Entwicklungsaufwand

Formate: JSON, XML, YAML, ...





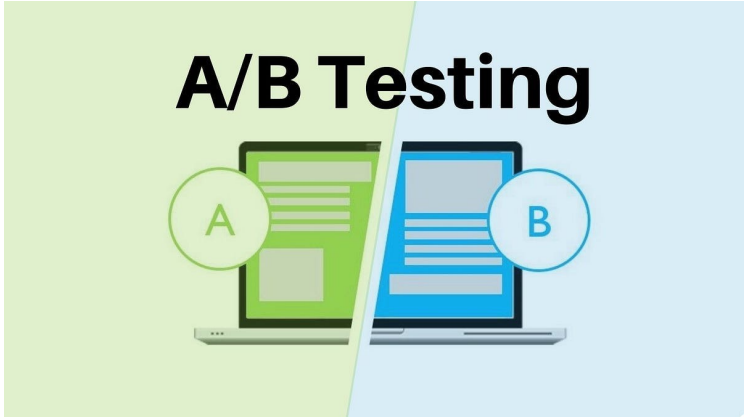
1. Schema definieren (Felder + Datentypen)
2. Schema im API-Aufruf übergeben
3. Sonderfälle / Edge Cases behandeln
4. Typische Daten und Beispiele verwenden

A/B-Testing von Prompts

Ziel: Bessere Prompt-Variante identifizieren
(höhere Relevanz/Genauigkeit/Zufriedenheit)

Kontrolle (A): Basis-Prompt

Variante (B): Optimierter Prompt



Metriken:

- Relevanz
- Vollständigkeit
- Verständlichkeit
- Stiltreue
- Wiederverwendbarkeit
- Effizienz

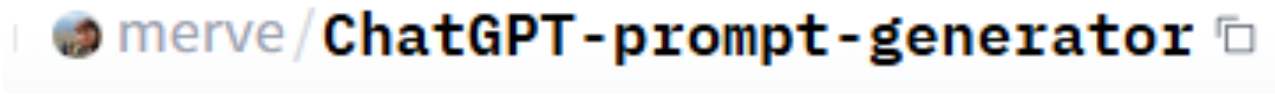
Testkriterien:

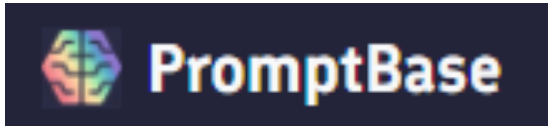
- Inhaltliche Tiefe
- Klarheit der Antwort
- Struktur & Format
- Zielgruppengerechtigkeit
- Nutzwert & Originalität

Prompt-Engineering-Tools

Management für größere Projekte oder in einem Team verwenden mit effizienter Verwaltung:

Tools für die Prompt-Entwicklung und -Optimierung:





“A prompt pattern is a structured template designed to guide and optimize user interactions with AI language models.”

Flipped Interaction Pattern

Purpose and Relevance

- Especially effective in areas with **little knowledge**
- Efficient Information Gathering
- **Inverts** User-ChatGPT **Interaction**
- ChatGPT drives conversation by asking goal-oriented **questions**
- ChatGPT may utilize knowledge absence to the user

Structure and Key Ideas

I would like you to ask me questions to achieve X

You should ask questions until this condition is met or to achieve this goal (alternatively forever).

(Optional) ask me the questions one at a time, two at a time, etc..

(Iterative) Instruction Pattern

Purpose and Relevance

- Useful for **repeatable processes**
- Describes processes **step-by-step** to ChatGPT
- ChatGPT executes steps as instructed
- Each step can build upon **input from previous steps**

Structure and Key Ideas

To create X, follow these steps:
Step 1: (Do Y). Ask me for Z.
I will provide A.
Step 2: etc.

(Optional) Ask me if I want to revise your output (then go to Step C) or if I want to start again (then go to step 1).

Now let's start with step 1.

Ihr Ansprechpartner

Dr. Jannik Fischbach

fischbach@fortiss.org

Gern stehe ich für einen weiteren Austausch zur Verfügung. Terminbuchung hier:


Many thanks.
Any questions?