Advertisements

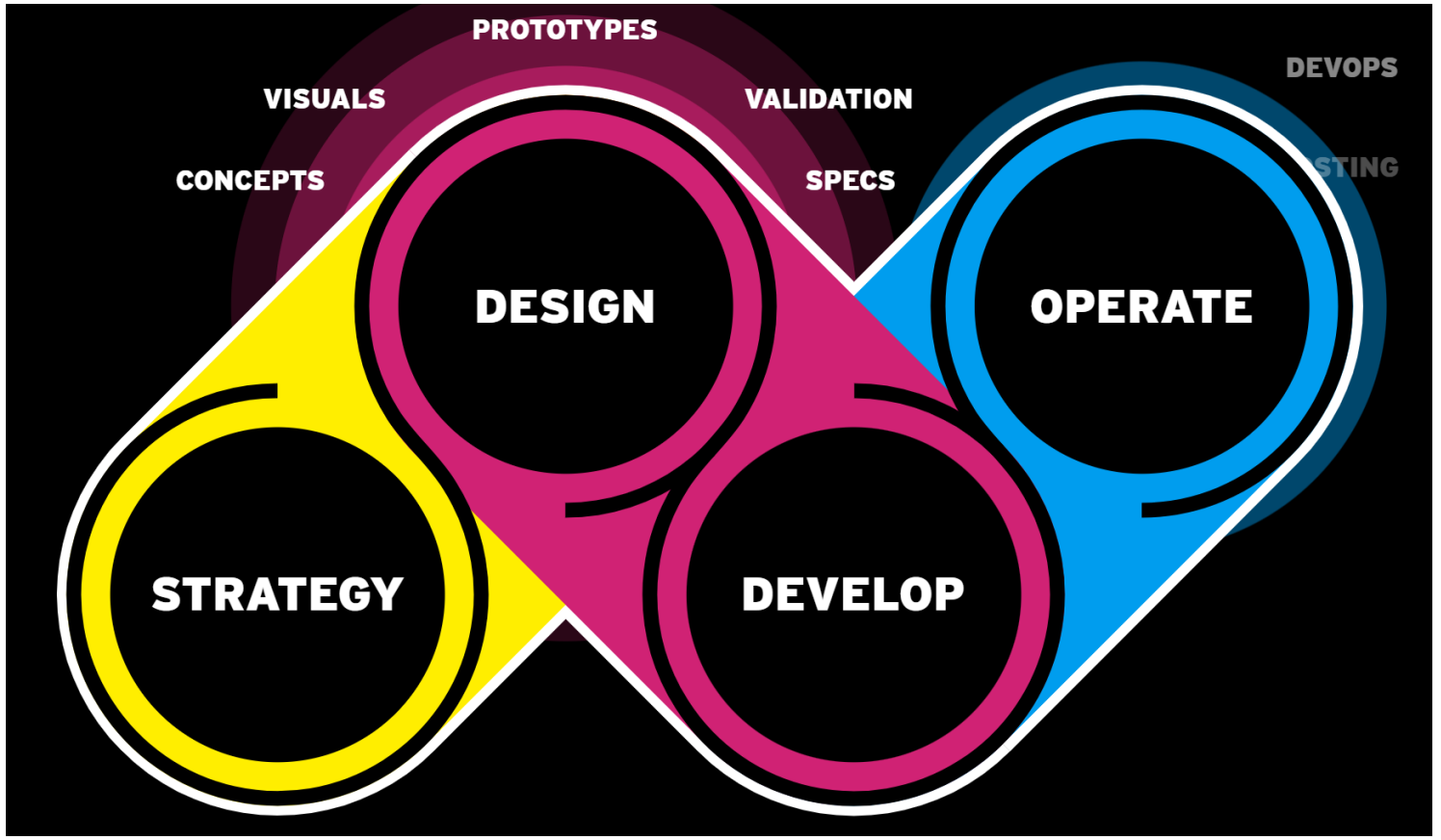
Collaborative Design: Guest Lecture via Zoom

Developing User-centered Products using Design Thinking and Collaborative Design

Stefan Schulz, ergosign

Monday, July 25th.

11:30 am



Seminar Software Quality

Seminar (B.Sc./M.Sc.)

Improve your understanding of Software Engineering by analyzing current research and practice of

- Test Impact Analysis
- Metrics for Software Sustainability
- Faster Testing in Go through Program Analysis
- Clone Detection
- Test Gap Analysis
- Flaky End-to-End Tests
- …

in **real** problems from a **real** industry partner: CQSE

Questions? Email: fabian.leinen@tum.de

Seminar: Inverse Transparency

Bachelor & Master Seminar

Do you want to explore research in **disciplines other than computer science**?

Do you want to solve **philosophical or societal issues** with your technical know-how?

Do you want to empower users with respect to **transparent data usage**?

Join our seminar!

Contact: julia.schuller+sit@tum.de

Requirements Engineering

Lecture 10

Prof. Dr. Alexander Pretschner

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

Orientation

Recap:

- RE in agile development, i.e. when change is expected

Coming up:

- Requirements Management
- NLP for analytical QA

Requirements Management and Quality Assurance

“How do I manage software requirements
if they change all the time?”

Definition Requirements Management (RM)

Requirements Management (RM) aims at:

- Management of the execution of the content (RE) tasks.
- Efficient and effective management and use of requirements throughout the system lifecycle.

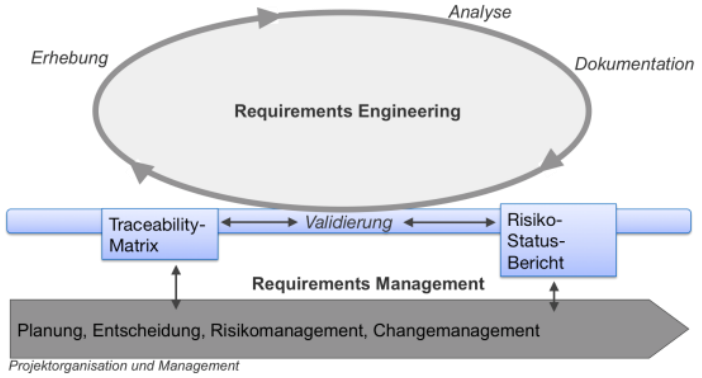
→ RM includes project organization tasks to efficiently support the implementation of requirements during the development process:

- Planning and cost estimation
- Risk assessment
- Quality assurance
- Modification / Change Management
- Claim Management

The tasks of the requirements management depend on the needs of the further phases in the project implementation.

Requirements Management: Core Tasks

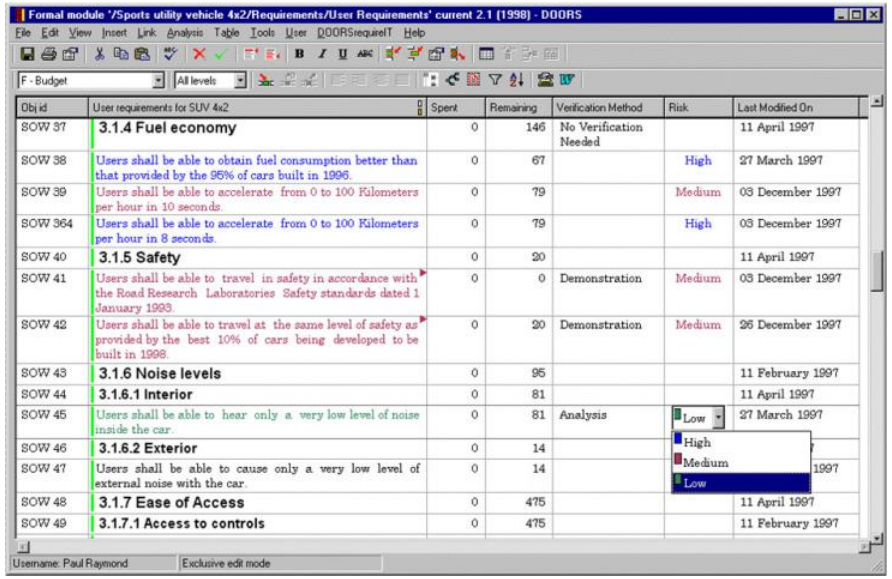
- Rationale Management and Traceability:
  - Justification of requirements
  - Linking requirements to each other and to development artifacts
- Requirements Management:
  - Structuring, documentation and archiving
  - Attribution
- Interactions with other management tasks:
  - Validation and verification
  - Change management including impact analysis
  - Version management
  - Configuration Management
  - Claim Management
  - Support for distributed RE



Manage Requirements

- Especially in large, distributed projects with several thousand requirements these requirements must be managed
- For this purpose, requirements
  - formally included in the list of considered requirements
  - managed via attribution
- At an appropriate time, requirements:
  - decided and fixed ("Freeze" / "Baseline")
  - summarized and approved

→ After their definition, the requirements are subjected to formal change procedures



Structuring and Attributing Requirements

An appropriate Structuring and Attribution of requirements are essential requirements for RM.

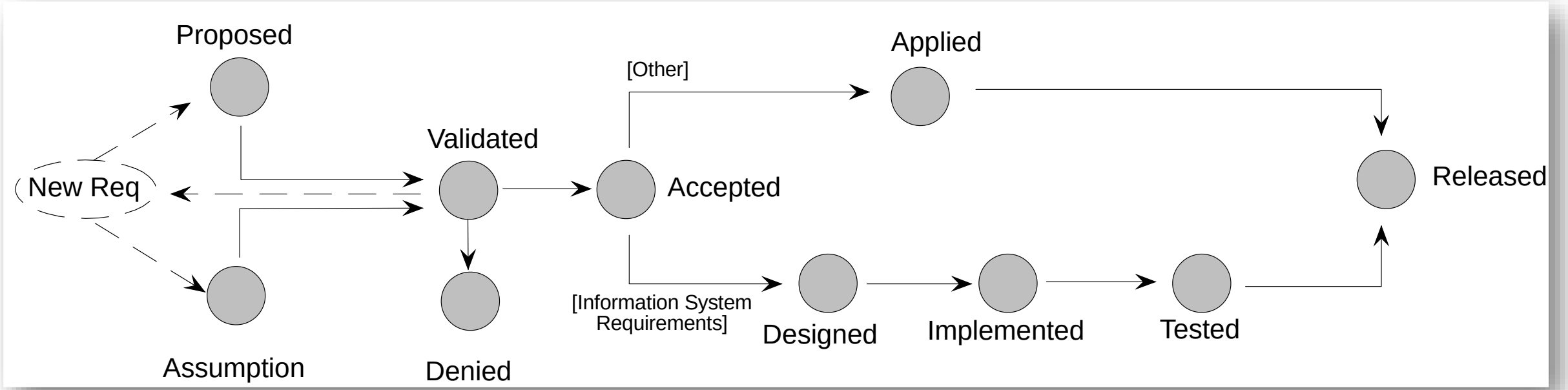
Example:

Attributes can represent the status of requirements, for example:
Requirements are approximate (in terms of the requirement life cycle):

- Proposed/Edited/Agreed/evaluated
- Accepted/Reserved/Rejected
- Validated
- Implemented
- Prioritize
- Changed/Replaced
- Verified

Status Model: Example from Quasar Requirements

Text



Elementary Attributes: Priority and Acceptance Criterion

Priority:

- Not always all requirements can be implemented and usually not all requirements are equally important

→ Prioritization of requirements, e.g., by means of Must/Could/Would ("MusCoW-Method")

→ Depending on the procedure in RE and the priority
  – requirements are implemented in steps (increments) and deleted (e.g., for time boxing)

Acceptance Criterion:

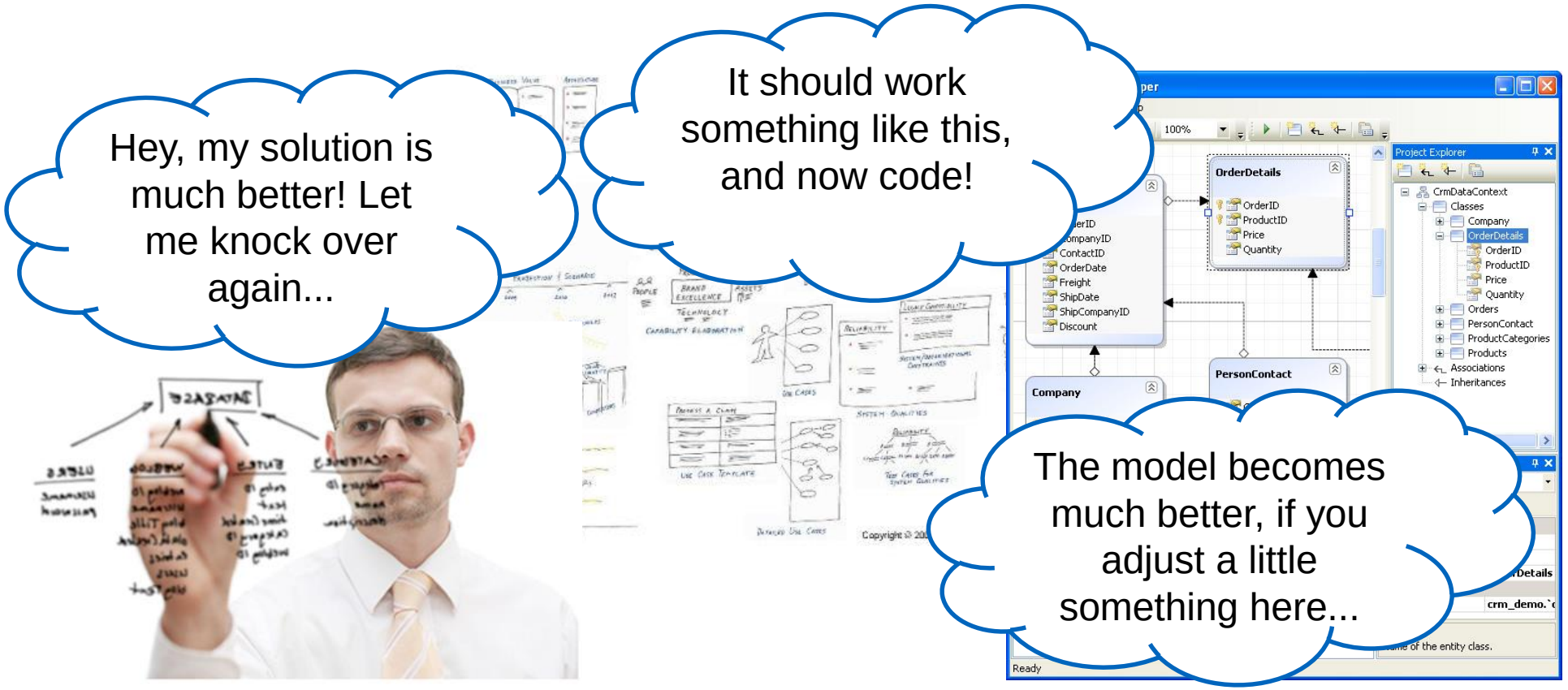
- It is not always possible to clearly evaluate and check all requirements, or to decide during acceptance when the implementation of the requirement was appropriate from the perspective of the stakeholders

→ Acceptance criterion defines criteria under which the implementation of the requirements is considered valid

Change Management

“Not just somehow, but with a clear process”

Formal Change Management - Why

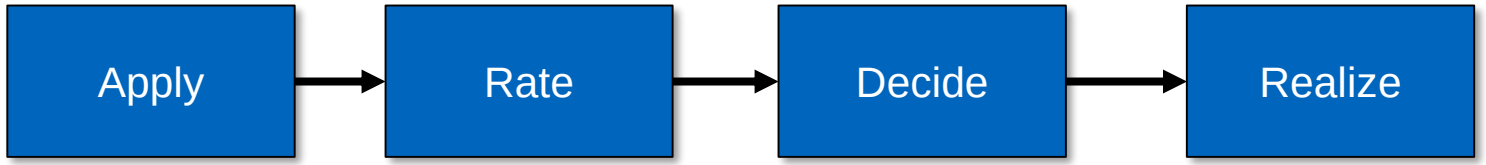


Internal: Avoiding uncontrolled growth in the development team
External: Avoid “Moving Target” projects

Change of Requirements

- Evaluation of change requests (sense and effort, risks etc.)
- Requirements and specifications may have to be changed even after they have been fixed during the project.
- **Reasons:**
  - Changes from inside (from development)
  - Changes from outside
  - One estimate: 3-10% requirement changes per month in business information systems

- Number and scope of changes determine the (in)stability of the requirements
- Procedure for handling amendments regarding requirements according to their specification:



Change Management

Objectives of change management for requirements:

- Systematic and controlled recording of change requests
- Evaluation of change requests (sense and effort, risks etc.)
- Decision on the implementation of the changes
- Implementation of the changes

Important related tasks:

- Impact analysis: What effects do changes have?
- Version management: Which versions of requirements and artifacts exist?
- Configuration management: Which requirements or artifacts together form consistent states?
- Error management: How are errors recorded, classified, documented, diagnosed, corrections determined and implemented?

Establishment of Change Management

- Define a change process
  - to record proposed changes (Change Request CR),
  - and to decide on their implementation
  - and to implement them.

- Create responsibilities and a change control board,
  - in which all interest groups are represented,
  - who is responsible for making decisions and monitoring the implementation of the changes

- Creation of a change database that supports the change process

Establishment of Change Management & Changes in RE

A Change Request (CR) includes

- what exactly should be changed and
- why it should be changed

For each CR:

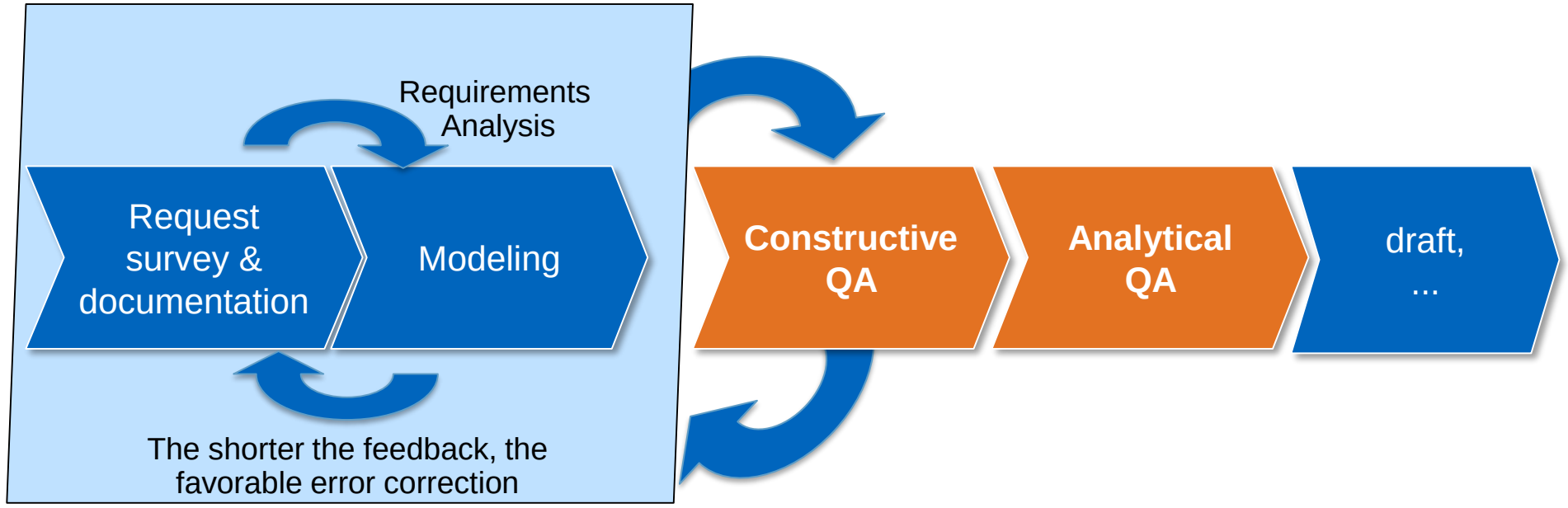
- perform an impact analysis to check which consequences/risks the change entails (with regard to affected artifacts).
- evaluate the benefits of the change, as well as the time and budget required for CR.
- to clarify who will bear the additional costs.
- implement the change and update the requirements documentation.

→ Requirement artifacts should be updated, changes logged (Interaction with problem management)

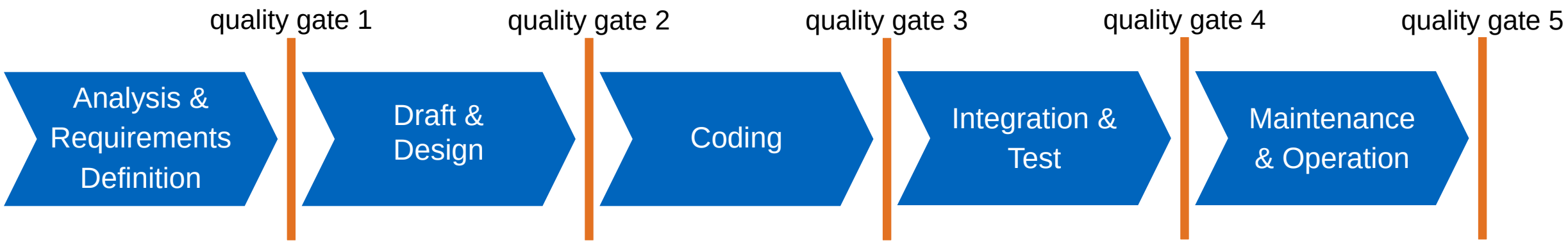
Analytical Quality Assurance (QA)

Quality Assurance in RE

- Constructive Quality Assurance: Assurance of the quality of the artifacts to be created during the creation process, e.g., by means of model building
- Analytical Quality Assurance: independent and autonomous testing and evaluation of the artifacts created, e.g., within the framework of quality gates



Quality Gates

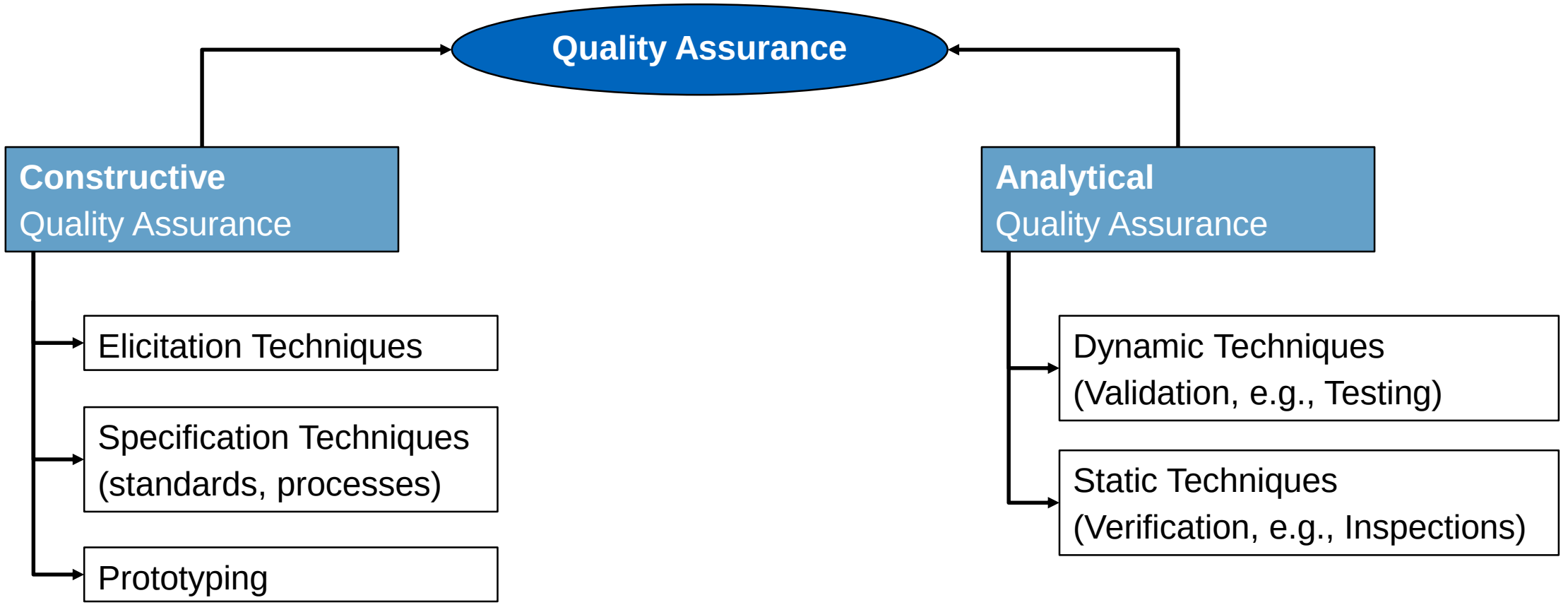


Quality Gates are quality measurement points

Different

- Test methods and objects, roles and times
- Test criteria and metrics

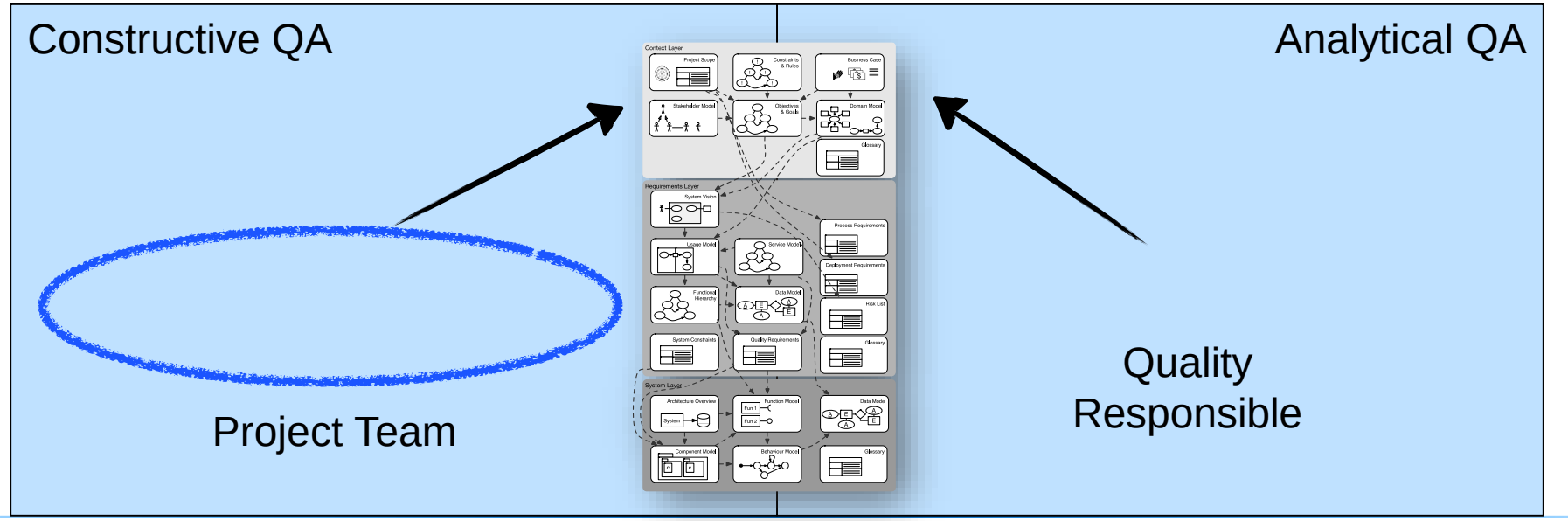
Overview of Quality Assurance (QA) in RE



Perspectives and Principles of Quality Assurance in RE

Selected quality criteria can and should only:

- Check by project members with domain knowledge during creation, e.g., "correctness", ...
  - framework: Constructive QA
- To be checked/evaluated by external/neutral responsible persons, e.g., "comprehensibility", ...
  - framework: Analytical QA



Terminology in the Context of Quality Assurance in RE

Important terms:

- Incorrect requirements: Requirement that do not reflect the intention of the stakeholders
- Quality deficiency: Requirement that may be valid in terms of content, but violates "quality criteria” e.g., lack of measurability, poor legibility, inconsistency, ...
- Interactions / critical consideration:
  - Incorrect requirements often hidden due to quality deficiencies
  - Correctness of requirements often considered as a quality criterion

Validation and Verification:

- Validation: Checking the requirements for correctness/validity
- Verification: Testing the system for compliance with the requirements

Quality Assurance:

→ Systematic measures to identify quality defects

→ Attention: Validation and verification are part of quality assurance

Smells in Requirements

- Smell detection is used for manual and automatic quality assurance
- Problem: Issues in requirements, such as ambiguities or incomplete requirements specifications, can lead to time and cost over-run
- Need to identify requirements defects fast and reliably
- Have seen examples of bad requirements before (really so bad?)
  → a sensor should work with *sufficient accuracy*: without detailing what sufficient means in the context, the specification is incomplete.
  → a certain property of the software under development should be fulfilled *as far as possible* can cause misinterpretations and difficult consequences during the acceptance phase of a product.
- This can efficiently happen through detecting smells

Smells in Requirements

- **Ambiguous Adverbs and Adjective:** e.g., almost always, significant, minimal.
- **Vague Pronouns:** e.g., “It should use the German layout”
- **Subjective Language:** e.g., user friendly, easy to use, cost effective
- **Comparative Phrases:** e.g., better than, higher quality
- **Superlatives:** e.g., best performance, lowest response time.
- **Negative Statements:** e.g., The system must *not* accept VISA credit cards.
- **Open-ended, Non-verifiable Term:** e.g., provide support, but not limited to, as a minimum
- **Loopholes:** e.g., if possible, as appropriate, as applicable
- **Incomplete References:** e.g., [1] “Unknown white paper”. Peter Mille

Natural Language Processing for Requirements

Why NLP? Common errors in specifications

Textual specifications are fundamental for SW development processes

Yet (remember user stories as an invite to converse)

- Often bad quality due to using natural language
- Problems: Inherent ambiguities, inconsistencies, etc.

Ideally, one fixes these errors at the beginning of the SW development process

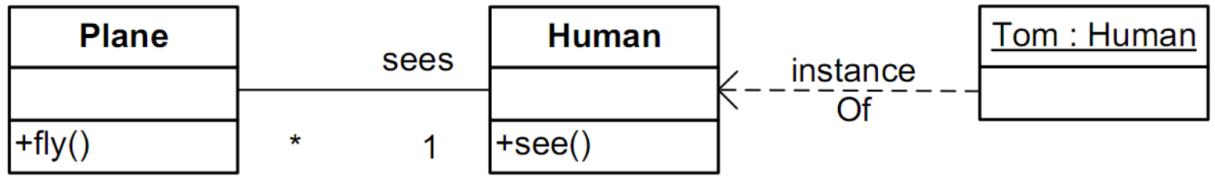
- Reviews, inspections, writing rules, etc.

Idea of Fabbrini et al.

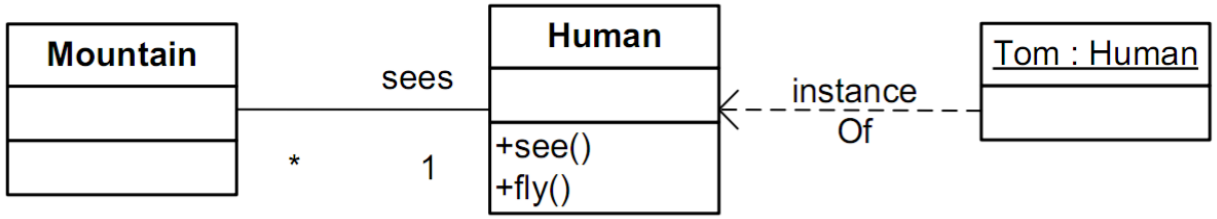
- There are quality attributes that are impossible/difficult to measure: unambiguity, completeness, consistency, etc.
- There are indicators for bad quality that can be found objectively

Why NLP? Common errors in specifications

Tom saw the plane flying.



Tom saw the mountains flying.



Quality model for specifications

Atomicity

- A specification entry is atomic if it cannot be usefully separated in more entries.

Unambiguity

A specification is unambiguous, if (a) all entries have the same meaning for all readers and (b) the specification includes no ambiguous words, phrases or sentences.

Conciseness

- A specification is concise, if (a) no unnecessary implementation details are described and (b) the specification contains no unnecessary entries or entries that could be formulated shorter.

Testability

- A specification is testable, if (a) each requirement has a method to check, if the system satisfies this requirement and (b) it is possible to derive test cases from the requirement.

Quality model for specifications

Traceability

- A specification is traceable, if (a) every reader can retrace the source and the further usage in the project life cycle of every requirement and (b) all redundant, interdependent and complementary information are set into dependency

Consistency

- A specification is consistent, if there are no overlaps in the content of requirements (a) in the specification, (b) between the specification and other relevant documents and (c) all terms are consistently used. Special cases for overlaps are conflicting or redundant requirements.

Formal Correctness

- A specification is formally correct, if (a) there are no linguistic defects (e.g. spelling, punctuation or grammatical mistakes) and (b) the specification is structured and supported in a way by graphical representations that every reader has the best possible support receiving information from the specification.

Quality model for specifications

Correctness

- A specification is correct, if (a) there are no terms, phrases or sentences in requirements with false content, (b) all requirements are at the right location, and (c) the specification reflects all currently valid requirements.

Completeness

- A specification is complete, if (a) every requirement is classified according to its importance, priority, necessity and liability, (b) if no requirements, parts of requirements or additional documents are missing, and (c) each possible quantitative information has been specified, and all terms are defined.

Common errors in specifications at Daimler

Quality-attribute	Distribution	
	#	%
Atomicity	52	1.03%
Unambiguity	54	1.07%
Conciseness	228	4.53%
Testability	3	0.06%
Traceability	274	5.44%
Consistency	433	8.60%
Formal Correctness	668	13.26%
Correctness	749	14.87%
Completeness	2575	51.13%
Total	5036	100.00%
Basis for analysis (defects)	5036	

Quality assurance with NLP: Lexical approach

Quality model of Fabbrini et al.
High-Level Properties

Non-Ambiguity

- the capability of each requirement to have a unique interpretation.

Specification Completeness

- the capability of each requirement to uniquely identify its object or subject.

Consistency

- the capability of the requirements to avoid potential or actual discrepancies.

Understandability

- the capability of each requirement to be fully understood when used for developing software and the capability of the requirement Specification Document to be fully understood when read by the user.

Quality model of Fabbrini et al.

Sentences containing the indicators are not incorrect in the sense of *wrong English* but in the sense of expressive power and comprehensibility!
The indicators can be determined automatically with the help of NLP tools

Non-Ambiguity: Indicators

Vagueness

- Sentence contains words with inherent vagueness (easy, fast, suitable, ...)
- Context size: sentence

Subjectivity

- Sentence references/builds on personal assessment or feeling.
- Context size: sentence

Optionality

- sentence allows choice.
- Context size: sentence

Weakness

- sentence contains weak verb.
- Context size: sentence

Specification Completeness: Indicator

Underspecification

- Sentence uses general noun that needs to be further specified.
- Context size: sentence

Consistency: Indicator

Underreference

- Sentence contains explicit reference to element not described.
- Context size: Entire document

Understandability: Indicators

Multiplicity

- Sentence contains more than one thing.
- Context size: sentence

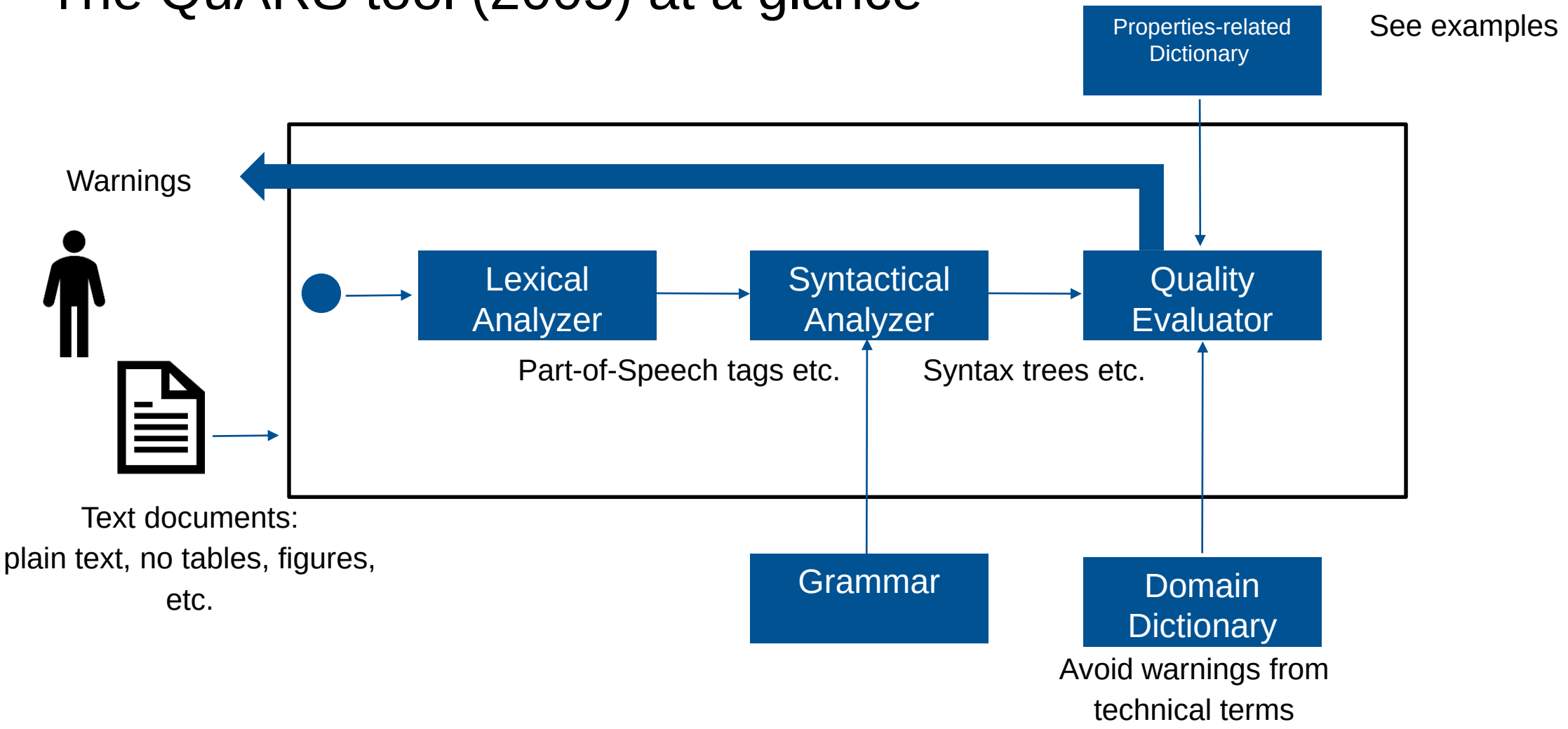
Implicitness

- Sentence subject is not explicitly named but general placeholder.
- Context size: sentence

Unexplanation

- Sentence contains undescribed/undefined acronym.
- Context size: Entire document

The QuARS tool (2005) at a glance



Detection examples

the C code shall be clearly commented

to the largest extent possible, the system shall be constituted by COTS products

the system shall be such that the mission can be pursued, possibly without performance degradation

the above requirements shall be verified by tests

Detection examples

Vagueness

- *the C code shall be clearly commented*

Subjectivity

- *to the largest extent possible, the system shall be constituted by COTS products*

Optionality

- *the system shall be such that the mission can be pursued, possibly without performance degradation*

Implicitness

- *the above requirements shall be verified by tests*

Detection examples

Vagueness

- *the C code shall be **clearly** commented*

Subjectivity

- ***to the largest extent possible**, the system shall be constituted by COTS products*

Optionality

- *the system shall be such that the mission can be pursued, **possibly** without performance degradation*

Implicitness

- *the **above** requirements shall be verified by tests*

Detection examples

the results of the initialization may be reported in a special file

the meantime needed to remove a faulty board and restore service shall be less than 30 minutes

the software shall be designed according to the rules of the Object Oriented Design

the handling of any received valid TC packet shall be started in less than 1 CUT

Detection examples

Weakness

- *the results of the initialization may be reported in a special file*

Multiplicity

- *the meantime needed to remove a faulty board and restore service shall be less than 30 minutes*

Underreference

- *the software shall be designed according to the rules of the Object Oriented Design*

No explanation

- *the handling of any received valid TC packet shall be started in less than 1 CUT*

Detection examples

Weakness

- *the results of the initialization **may be** reported in a special file*

Multiplicity

- *the meantime needed to **remove** a faulty board **and restore** service shall be less than 30 minutes*

Underreference

- *the software shall be designed according to the rules of the **Object Oriented Design***

No explanation

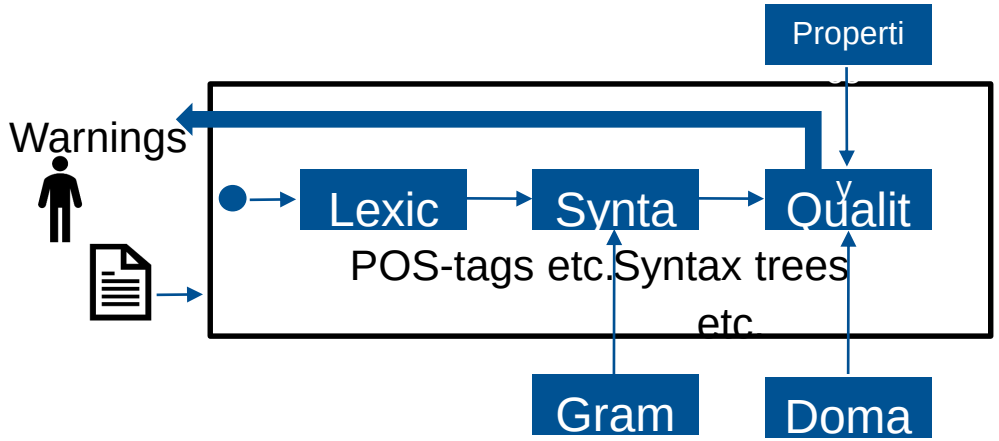
- *the handling of any received valid **TC** packet shall be started in less than 1 **CUT***

Discussion: What should we change if we wanted to ...

… change the domain, e.g., automotive to avionics?

… use another language, e.g., German instead of English?

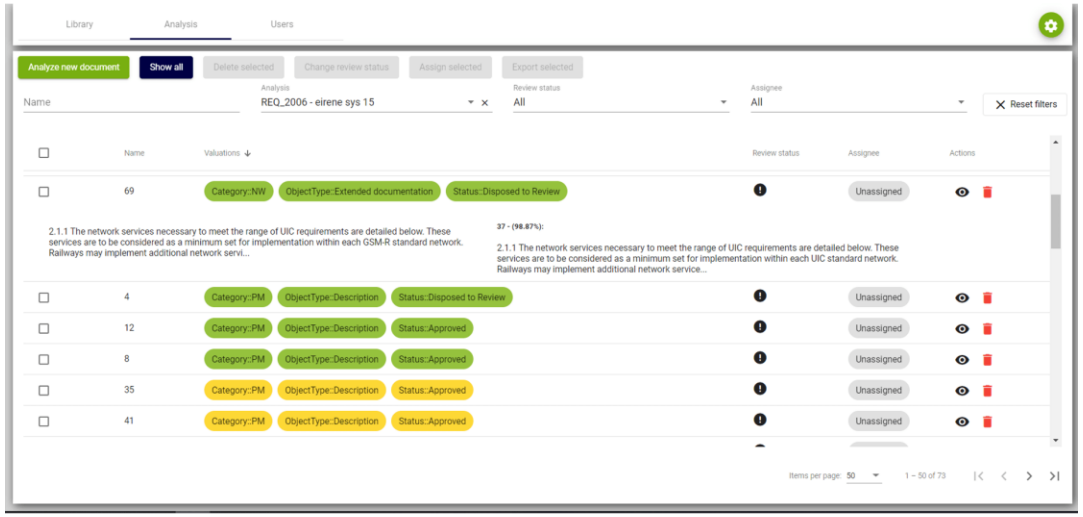
… other indicators?



NLP-based RE: State of the Art

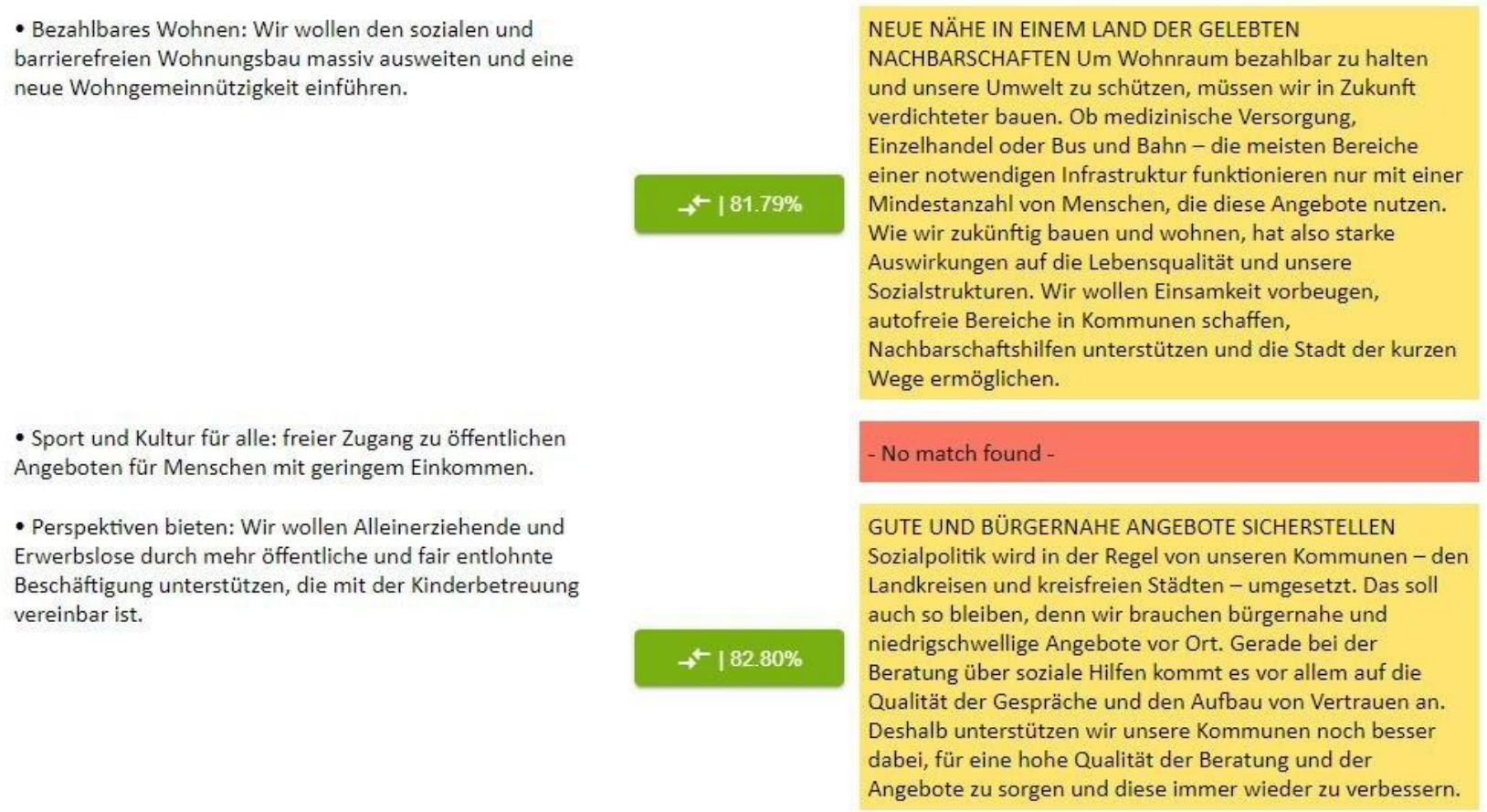
Semantha

Use already evaluated specifications to classify new requirements
Use ontologies to find references to external documents and link them
Find risks, e.g., inconsistencies, within the documents



https://www.semantha.de

Example classification: party programs (Die Linke vs. SPD)



References

D. Ott, "Defects in natural language requirement specifications at Mercedes-Benz: An investigation using a combination of legacy data and expert opinion," 2012 20th IEEE International Requirements Engineering Conference (RE), 2012, pp. 291-296, doi: 10.1109/RE.2012.6345817.

F. Fabbrini, M. Fusani, S. Gnesi and G. Lami, "The linguistic approach to the natural language requirements quality: benefit of the use of an automatic tool," Proceedings 26th Annual NASA Goddard Software Engineering Workshop, 2001, pp. 97-105, doi: 10.1109/SEW.2001.992662.

S. J. Korner and T. Brumm, "RESI - A Natural Language Specification Improver," 2009 IEEE International Conference on Semantic Computing, 2009, pp. 1-8, doi: 10.1109/ICSC.2009.47.

Summary

- Requirements Management of central importance for
  - Management of the execution of the content (RE) tasks.
  - Efficient and effective management and use of requirements throughout the system lifecycle.

- Attribution and traceability support a wide range of management tasks accompanying the overall project implementation:
  - change management
  - risk management

- Outlook: NLP technology to detect linguistic deficiencies

Outline and Outlook

Terms and Definitions
Context-specific nature of SE and RE
Quality models for requirements
Engineering Models
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