Requirements Engineering

Lecture 3

Prof. Dr. Alexander Pretschner

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

Orientation

Recap:

- Many definitions
- Core activities in RE
- Quality model for requirements

Coming up:

- Where do we start and what do we do, to create a useful specification?
- Two different philosophies to approach RE tasks

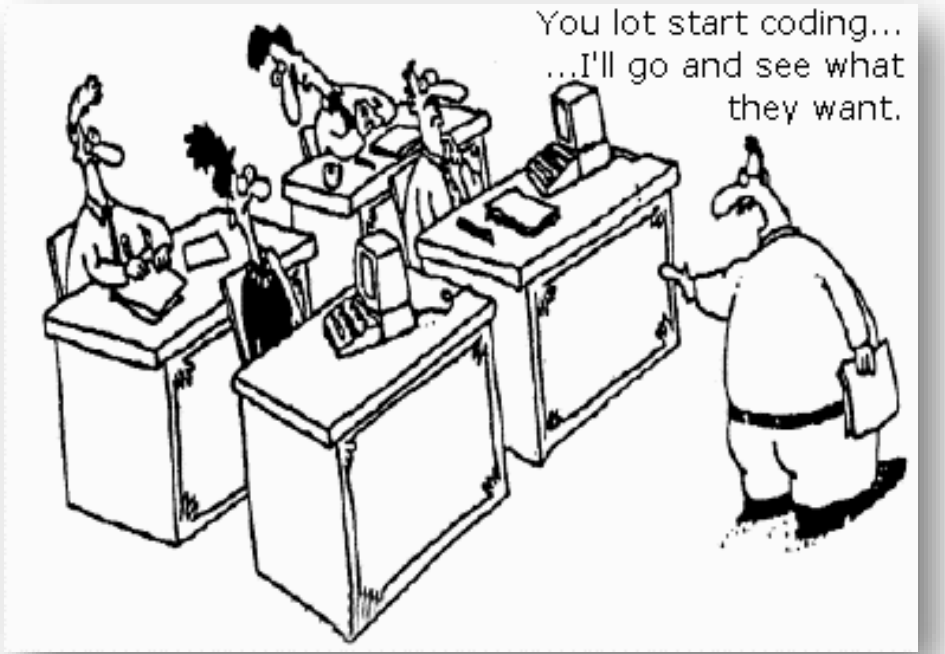
Explicit versus implicit RE

RE as an **implicit task**

- "*It is the code that counts*"
- Clarify requirements during development

RE as an **explicit task**

- Requirements must be clarified by the responsible persons so thoroughly that no decisions related to the requirements are made by programmers during the implementation



Picture taken from: http://acceleratedevelopment.ca/blog_images/You_lot_start_coding.gif

Explicit Requirement processes according to ISO/IEEC 29148

6.1 Requirement processes

The project shall implement the following requirements engineering processes as defined in ISO/IEC/IEEE 15288 and ISO/IEC/IEEE 12207, depending on the adherence to one or both of ISO/IEC/IEEE 15288 and ISO/IEC/IEEE 12207.

— **Business or mission analysis process** (ISO/IEC/IEEE 15288:2015, 6.4.1 or ISO/IEC/IEEE 12207:2017, 6.4.1).

— **Stakeholder needs and requirements definition process** (ISO/IEC/IEEE 15288:2015, 6.4.2 or ISO/IEC/IEEE 12207:2017, 6.4.2)

— **System requirements definition process** (ISO/IEC/IEEE 15288:2015, 6.4.3) or System/software requirements definition process (ISO/IEC/IEEE 12207:2017, 6.4.3)

Problem: Standards are (deliberately) vague and hard to apply to a specific problem.

(Explicit) Requirements Engineering

RE represents the bridge from stakeholder goals into software development.

- RE is a decision-making process:
  - Recognize, evaluate, decide.
  - RE is part of the decision whether a project should be carried out.

- RE must always start with the individual project starting point
  - Are goals clear - is there agreement on them?
  - Are responsible persons RE-capable?
  - Is the application domain understood?

→There cannot be one standardized process for all projects.

Essential aspects in explicit RE

Are roles ("responsibilities") clearly defined?

Is the engineering procedure clearly defined?

- Are requirements explicitly developed and documented?
- Are requirements validated?
- Is the quality of the requirements checked?
- Is the correct implementation of the requirements systematically monitored?
- Is there an updated description of the implemented requirements at the end of the project?

→In other words: What is the **engineering model** used in this project?

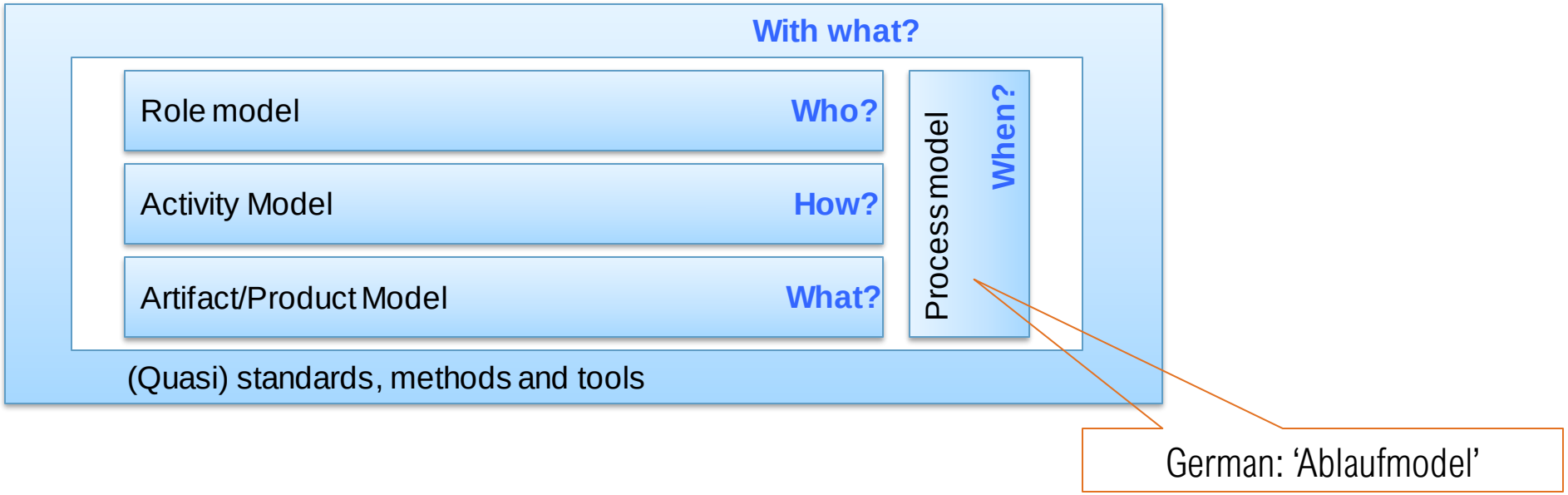
German: 'Vorgehensmodell'

What is an engineering model?

Definition

An engineering model describes systematic, engineering-like and quantifiable procedures to solve tasks of a certain class in a repeatable way.

Submodels

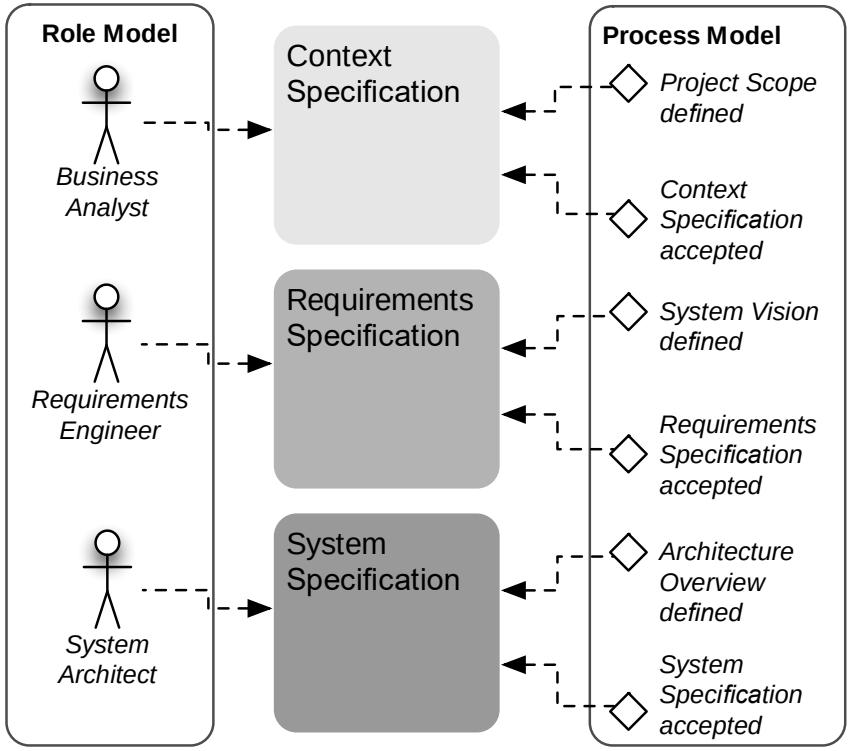


Role model and process model

Process model: Instantiation at the beginning of the project defines milestones

Role model:

- Roles have explicit skills and the responsibility for an artifact
- Example are:
  - Business analyst as domain expert
  - Requirements Engineer as mediator between context areas and development
  - System Architect
- Roles can be filled by different people or by the same person in personal union



Engineering Models: Two schools of thought

Activity-orientation

- Focus on the **activities** performed
- Specifies who does what in which way and at what time

„First, an initial set of stakeholders is identified in a brainstorming session using a whiteboard. Then, colors are used to highlight stakeholders that share a common interest…“

Artifact-orientation

- Focus on **what** is created and its characteristics
- Defines responsibilities per artifact

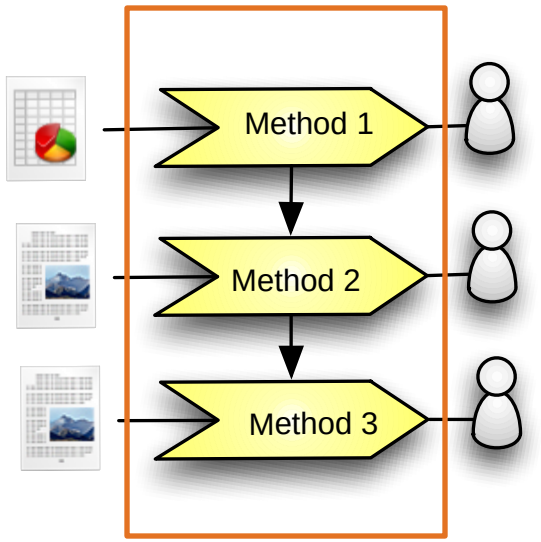
- *„The stakeholder model captures the individuals, groups or institutions having responsibility in the projects as well as their relationships and hierarchy.“*

Exercise: Which approach did you and your advisor follow during your Bachelor's thesis?

Activity-oriented approach

How is the project approached?

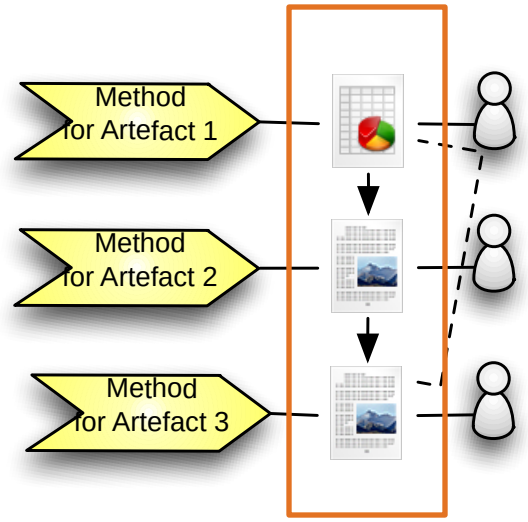
- Roles carry out activities
- Artifacts serve as input/output



Evaluation:

Advantages	Disadvantages
Description of the work process	Restrictive: This way and only this way!
Specification of a temporal order and detailed instructions for action	Complex planning, difficult to measure the quality of the results
	No statements about artifact contents and dependencies

Artifact-oriented approach



What is developed/created in the project?

- Roles are responsible for the artifacts or support
- Activities serve exclusively for planning and result generation

Evaluation:

Advantages	Disadvantages
Awareness of clear result structures (content, dependencies, terminology)	High learning curve (thinking in processes is habitual behavior)
Testable quality and progress control	Definition & selection of adequate methods
Clear roles and responsibilities	Derivation of plans complex
Good adaptability	

What is an artifact model?

Artifact:

- Documents (intermediate) results of development process steps
- Has structure, content and a form of representation
- Has sense and purpose, subject to version control
- Examples: Documents, data objects and models

Artifact model:

- All artifacts and dependencies relevant in the process
- In the literature there are numerous definitions and terms that are often used synonymously (meta model, ontology, product model, …)
- Artifact models can be constructed and displayed in different ways!

Why artifact orientation in RE?

Common problems in practice	Characteristics of artefact orientation
Maintenance hard w/o solid requirements documentation (but also: write-only-documents)	Documentation
Roles, responsibilities and skill profiles often unclear	Definition of responsibilities along the created artifacts
Lack of understanding of content, context and terminology in the artifacts	Implicit knowledge about (domain) content is explicitly recorded ▪ Modelling concepts and notations ▪ Contexts → Awareness for consistent and complete results → Integration into the QA process (testable result structures)

A Note on Artifacts

Agile development procedure skeptical about number and level of detail of artifacts beyond code and tests: **no immediate customer value**; hard to maintain over time (will discuss agility later)

Artifacts need to be **synchronized** with each other

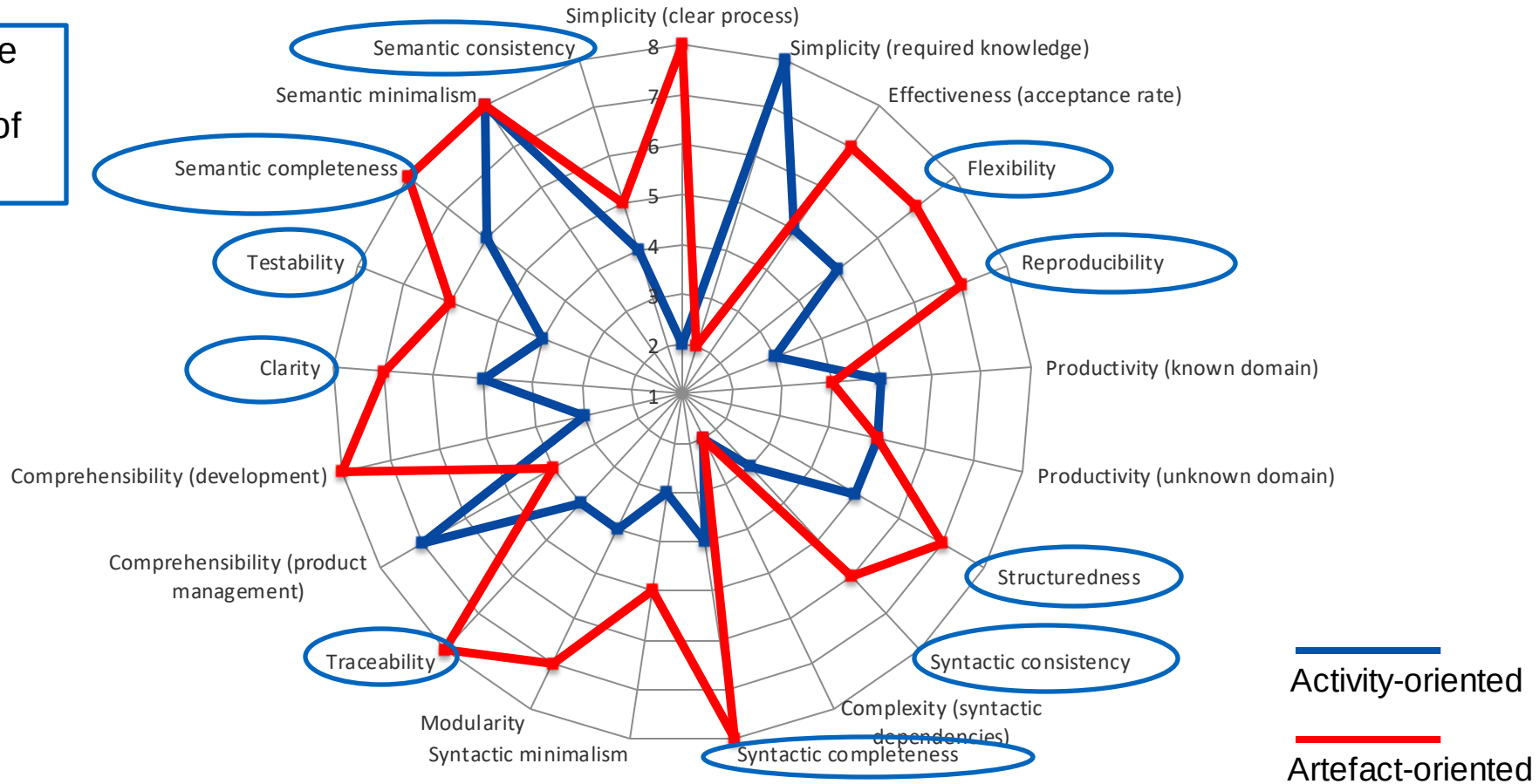
Change of context/requirement: **cost of changing artifacts**

"Quality" of artifacts in general hard/impossible to assess! This is the reason for standardized processes: ISO26262 functional safety; Common Criteria for security; ISO 9000 for quality in general. Require existence of processes and certain artifacts but do not state their quality.

However, **maintenance is difficult without well-documented requirements**

Empirical studies (advantages)

Artifacts increase syntactic and semantic quality of requirements

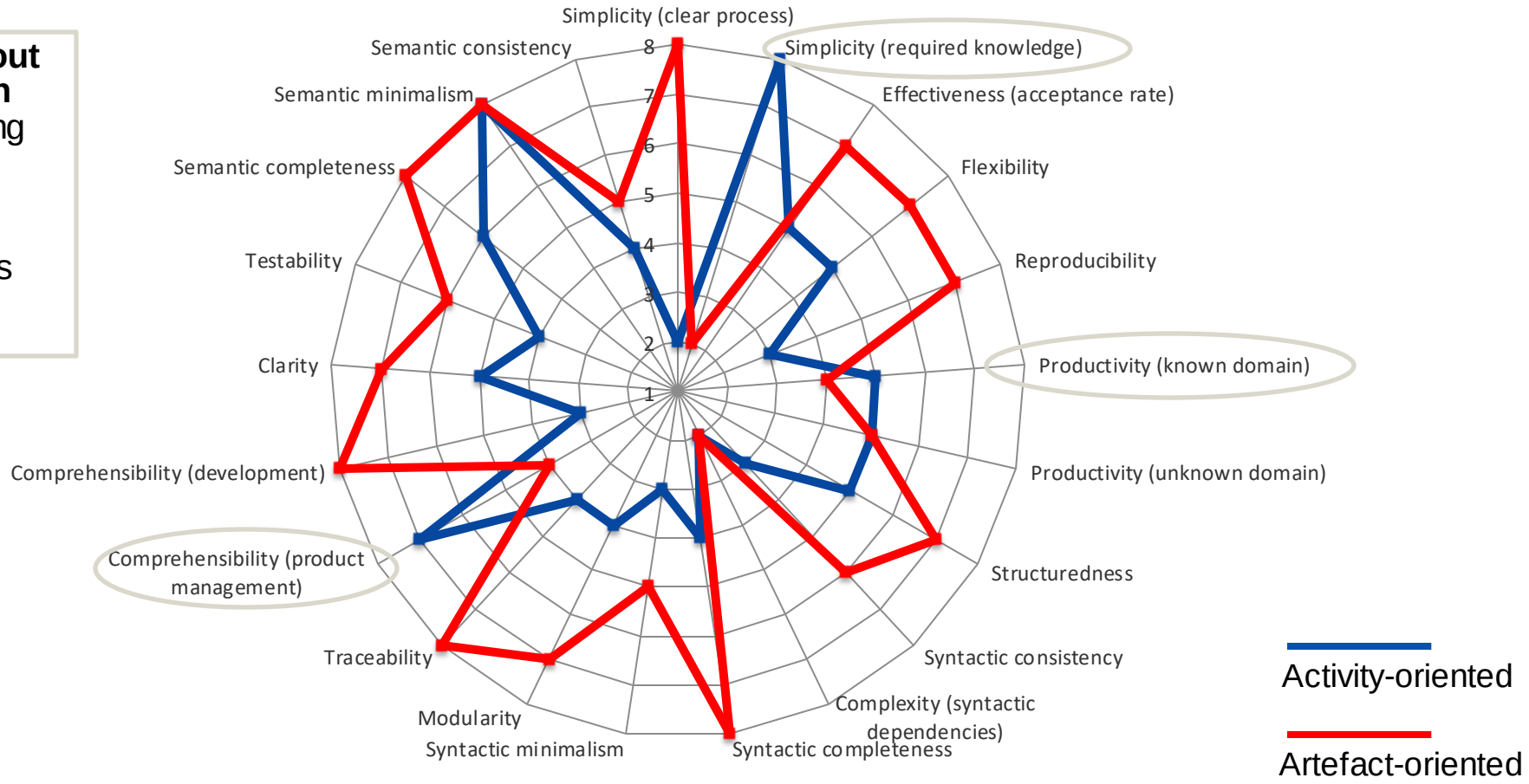


* Mendez et al.. A Case Study on the Application of an Artefact-Based Requirements Engineering Approach (EASE 2011).

Empirical investigations (disadvantages)

Critical points about artifact-orientation

- Requires learning curve
- Productivity?
- Choice of notions/concepts essential



* Mendez et al.. A Case Study on the Application of an Artefact-Based Requirements Engineering Approach (EASE 2011).

Recap: Experience in the use of artifact models

Use of artifact models:

- Advantages:
  - Flexibility in the process and freedom for creativity and use of discretionary powers in the procedure
  - Precise, testable artifacts/specifications
  - Clear terminology across project boundaries
- Disadvantages:
  - Artefact orientation requires learning curves
  - The choice of notation/models is essential for the creation of artifacts
  - Artifacts and agility!

Construction and further development of artifact models

- Introduction of a new paradigm requires support from management
- Construction and integration of an artifact model (depending on complexity) very complex:
  →Example Capgemini TS (analysis, conception, transfer to tools, process integration and training): 112 PM

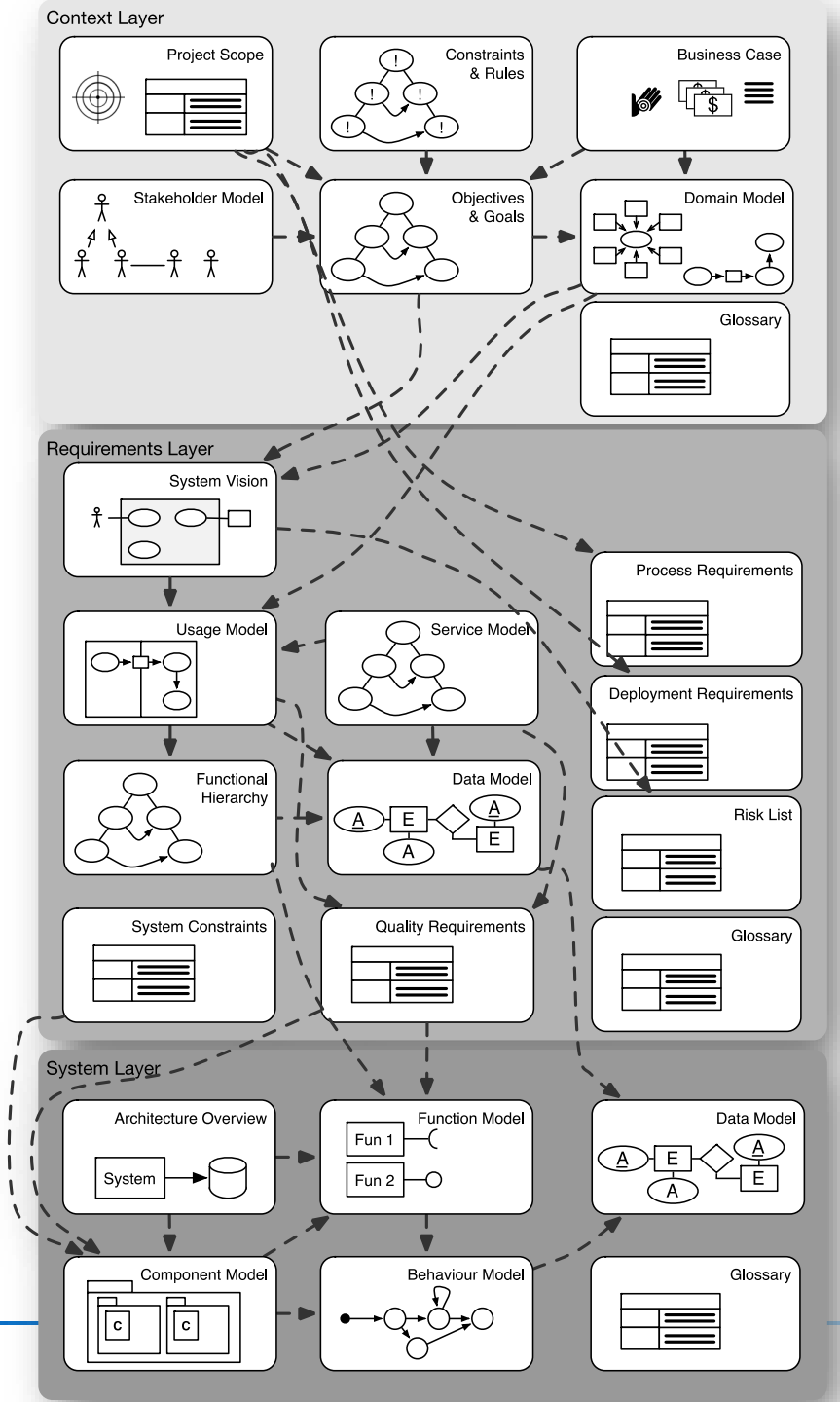
→Further development and modification of an artifact model is costly, because the content model is partly very complex and depends on hundreds of elements

AMDiRE artifact model

AMDiRE:

Artefact **M**odel for **D**omain-**i**ndependent **RE**

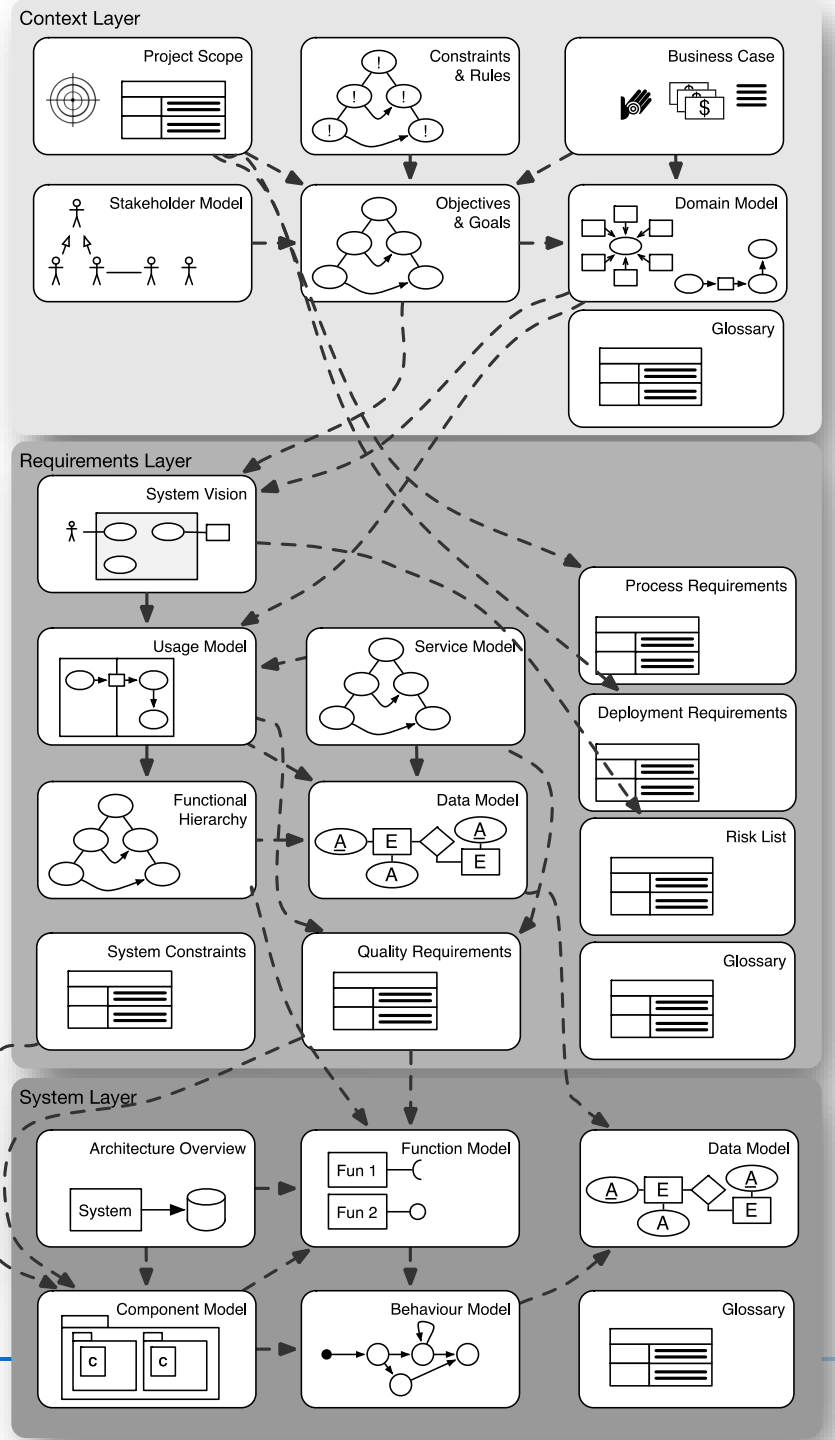
An example for an artifact-oriented engineering model created at TUM



AMDiRE artifact model

The flexibility of an artifact model is rooted in the selection of adequate artifacts.

Which artifacts would you prepare for the development of the user frontend in our "Rent-a-Scooter" system? Which ones for the on-board software?



AMDiRE: Roles and Layers

Business Analysts /
Domain Expert

Req. Engineer

System Architect /
Developer

Problem	*Context Layer* *Need, Business Req.*
Requirements	*Requirements Layer*
Solution	*System Layer* *System Requirements*

Problem space vs. solution space

In Requirements Engineering, we must capture both the problem and solution space.

Problem

1

- Capture stakeholders and their goals
- Understand and describe the problem
- Specify and validate characteristics and capabilities of potential solutions

Solution

2

- Iteratively derive a specification of a solution, that solves the problem
- Develop function and architecture models
- Establish tracking and verification
- Incorporate changes

The distinction between problem definition and solution is essential in RE!

Engineers are trained to focus on the solution!!!

Separating Problem and Solutions is not trivial!

Common examples for solution orientation:

- *„The goal of this project is to build an app, that…“* (German Corona Warn App)
- *„Where can I rent a car to drive to the supermarket to get groceries?“*
- *„To avoid that our server crashes when all students submit their homework five minutes before the deadline, we will need a more powerful server.“*

In contrast: „Design Thinking is about understanding the problem first”

What is wrong with (premature) solution orientation?

Requirements are often not explicitly identified, but it is rushed to a solution.

- Essential requirements are not considered
- Solution space is unnecessarily restricted
- Decisions are not validated with stakeholders
- System has properties, which are not required ("Gold Plating") and do not add immediate value (*“Wouldn’t it be cool to have this feature as well?”*)

Outline and Outlook

Terms and Definitions
Core Activities
Quality Model for Requirements

Engineering-Models: artefact-orientation vs activity-orientation
Separation of Problem and Solution

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