UNIVERSITÄT DUISBURG ESSEN

*Offen im Denken*

Requirements Engineering & Management

# Introduction and Fundamentals II

Prof. Dr. Klaus Pohl

© SSE, Prof. Dr. Klaus Pohl

PALUNO
The Ruhr Institute for Software Technology

# Agenda



1. Impact of Constraints
2. Non-functional Requirements
3. “What?” vs. “How?”
4. Requirements Engineering as an Early Development Phase
5. Requirements Engineering as a Cross-Cutting Activity
6. Requirements Engineering as a Continuous Process

© SSE, Prof. Dr. Klaus Pohl

# Impact of Constraints

© SSE, Prof. Dr. Klaus Pohl

# Impact of Constraints



## ... on the Development Process:

C-16: The effort for system development shall not

exceed 480 person months.

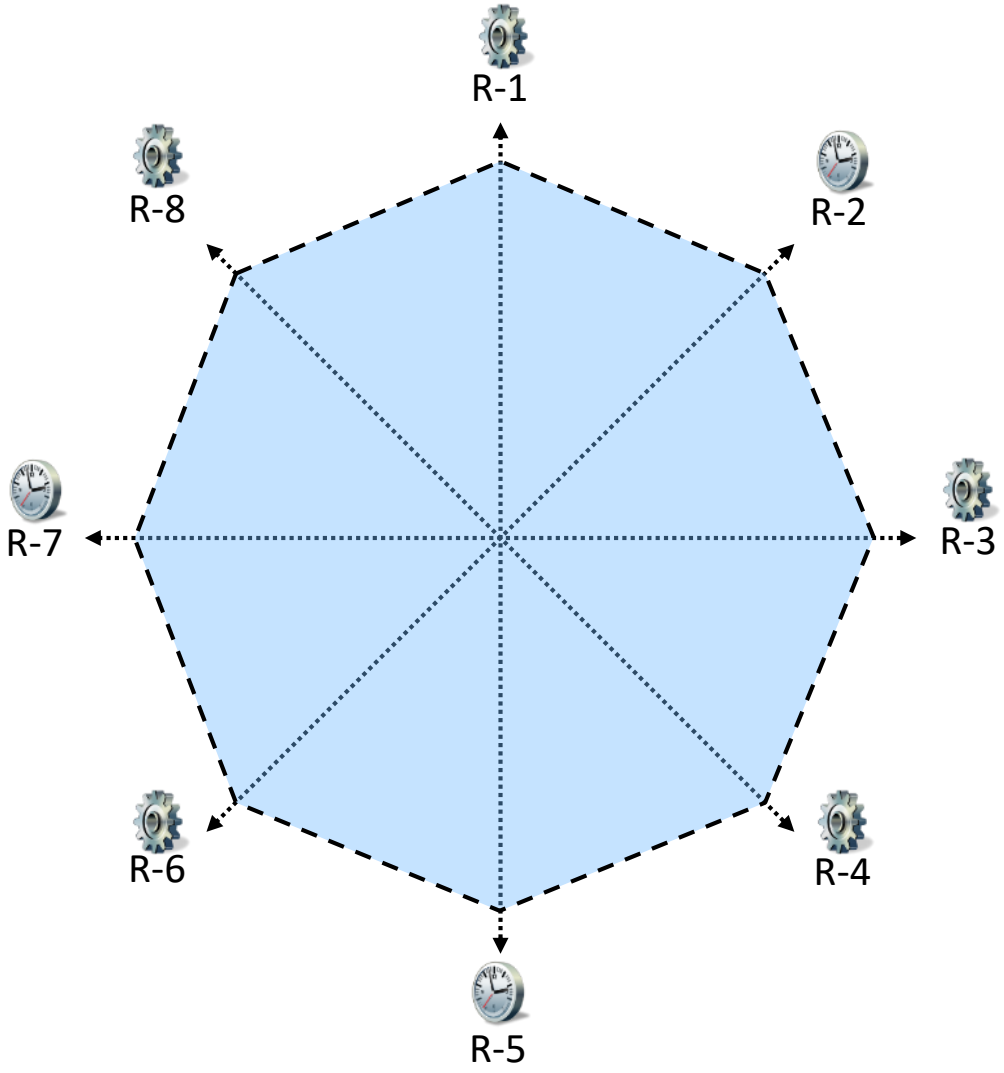
## ... on the System:

C-36: The electronic control unit in the vehicle interior shall work at temperatures from -10 to +50 degrees Celsius.

Most constraints influence both the development process and the system itself (i.e. the properties of the system).

© SSE, Prof. Dr. Klaus Pohl

# Restrictions imposed by Constraints (1)




Valid realisation alternatives for requirements without constraints

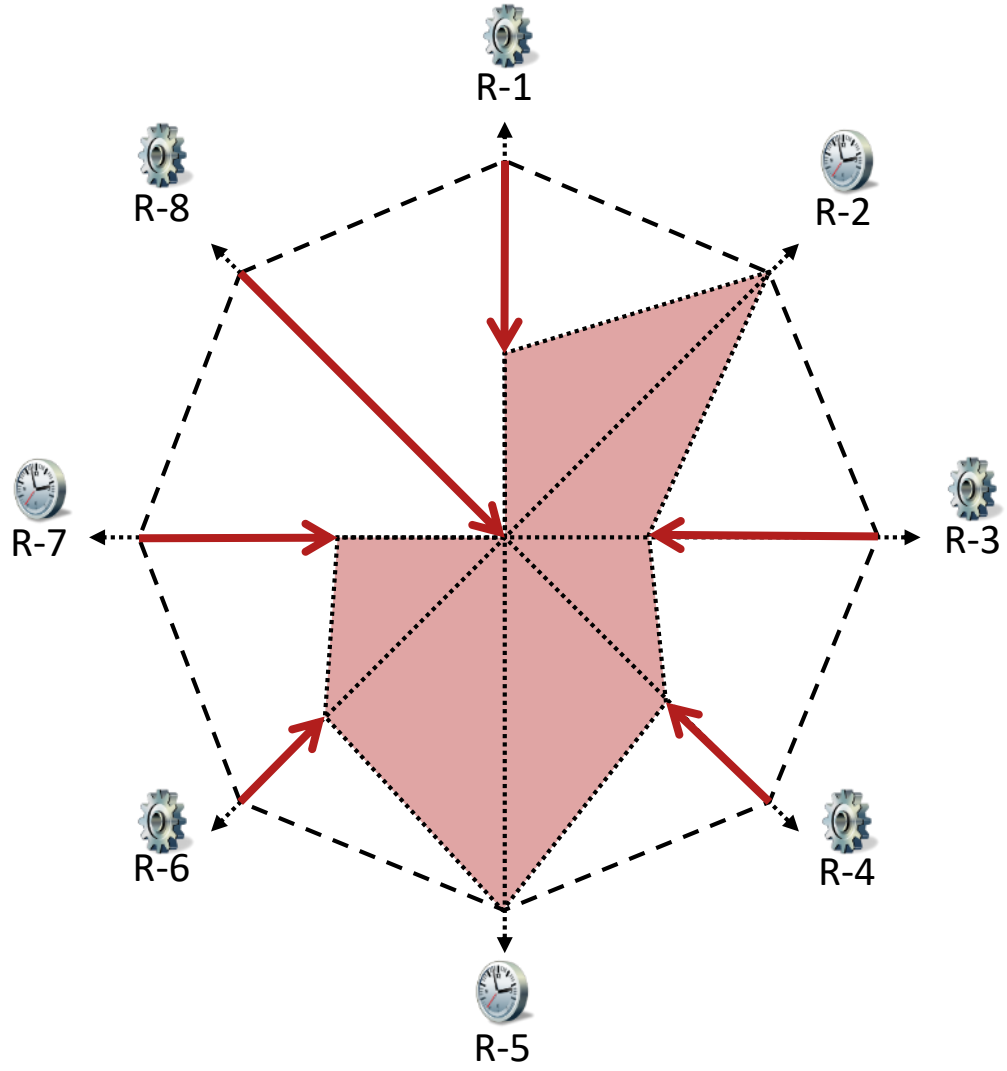


All icons from [1]



© SSE, Prof. Dr. Klaus Pohl

# Restrictions imposed by Constraints (2)




Valid realisation alternatives for requirements without constraints

Reduced realisation alternatives for requirements caused by constraints

Reduction of valid realisation alternatives for a requirement caused by constraints

All icons from [1]

© SSE, Prof. Dr. Klaus Pohl

# Restrictions imposed by Constraints (3)



R-3: The output shall be presented on a mobile phone.

Possible solutions include:

iOS

Tizen

Ubuntu Touch

Android



© SSE, Prof. Dr. Klaus Pohl

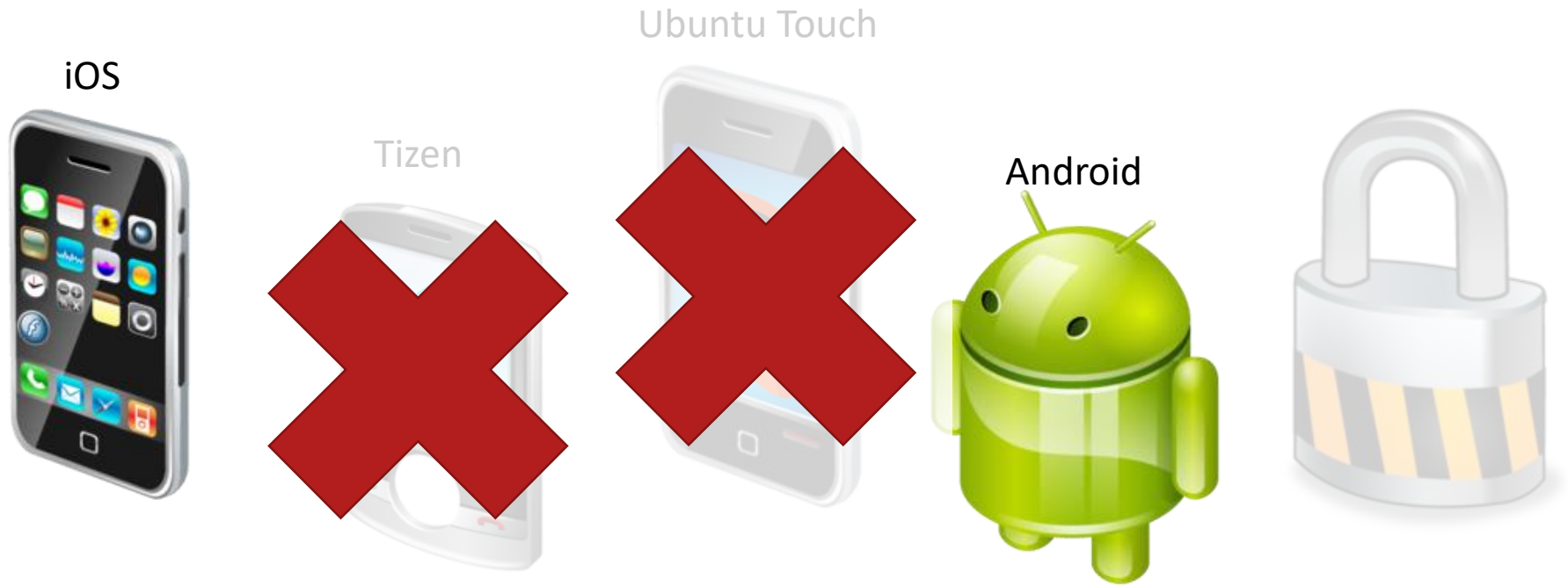
Icons from [1, 3, 4]

# Restrictions imposed by Constraints (3)



C-4: Only iOS and Android shall be supported.

**Consequence**: 2 out of 4 initial solutions left (50%)




Icons from [1, 3, 4]

© SSE, Prof. Dr. Klaus Pohl

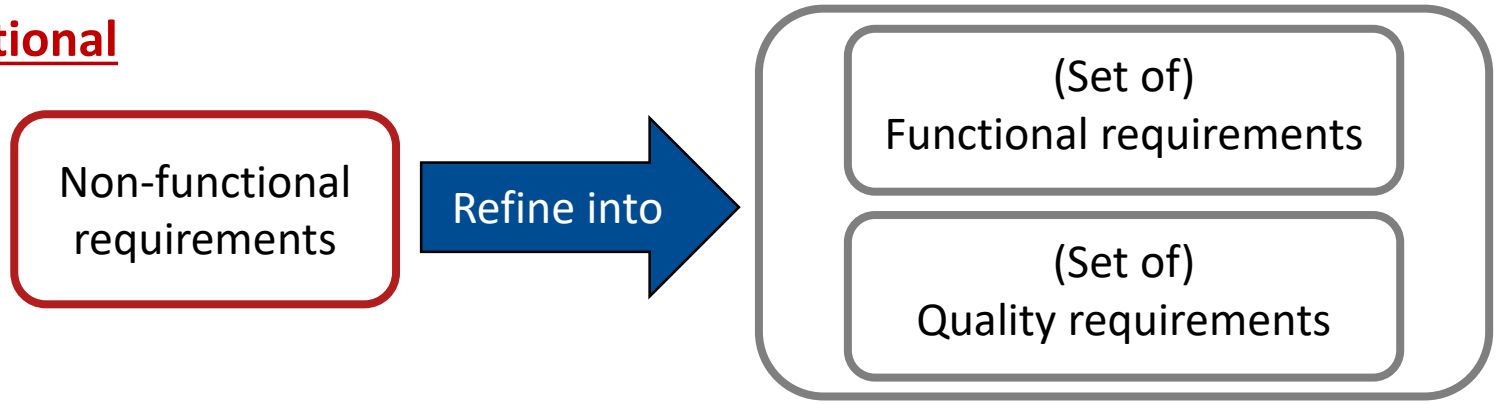
# Non-functional Requirements

© SSE, Prof. Dr. Klaus Pohl

# Non-functional Requirements



- The term “non-functional requirement” is **widely used** in practice and literature.
- Whenever a requirement is called non-functional requirement, the requirement itself is typically **insufficiently understood**.
- However, non-functional requirements are often **accepted just because they are categorised as non-functional**.
- Most of the non-functional requirements are either
  - **Underspecified functional requirements**,
  - Underspecified quality requirements, or
  - Both **underspecified functional** and **quality requirements**.




© SSE, Prof. Dr. Klaus Pohl

# Example of a “Non-functional“ Requirement (1)



NFR-12: The system shall be secure.

**This is clearly an underspecified requirement**. It raises, among others, the following questions:

- What does the adjective “secure” **mean**?
- Which **properties** shall the system provide in order to be “secure”?
- Which **functionality** offered by the system should be “secure”?
- Which **data the system deals with** should be “secure”?
- How can one **check whether the implemented system is “secure”**?

© SSE, Prof. Dr. Klaus Pohl

# Example of a “Non-functional“ Requirement (2)



NFR-12: The system shall be secure.

This underspecified requirement should be **refined into more precise functional and quality requirements**, e.g.:

R-12.1: Each user shall log in to the system with his user name and password prior to using the system.

R-12.2: The system shall remind the user every four weeks to change the password.

R-12.3: When the user changes his password, the system shall validate that the new password is at least eight characters long and contains alphanumeric characters.

R-12.4: The user password stored in the system shall be protected against password theft.

© SSE, Prof. Dr. Klaus Pohl

# “What?” vs. “How?”

© SSE, Prof. Dr. Klaus Pohl

# Problem ("What?") vs. Solution ("How?")



- **Problem statement** describes "**what**" shall be achieved
- **Solution description** defines "**how**" that problem shall be addressed
- **Guiding principle** to tackle **complex problems**: **Differentiation** between the questions "**what is the problem?**" and "**how can it be solved?**"
- Separation between "what" and "how" in requirements.
  - Requirements can be defined at different levels of abstraction → **multilevel hierarchy of requirements**
  - Problem: Stated by upper-level requirement
  - Solution: Proposed by the lower requirements

based on [Ward and Mellor 1985]

© SSE, Prof. Dr. Klaus Pohl

# In Requirements Engineering: Refinement of Requirements



- Underspecified requirement: Describes the **"what" at an abstract level**
- The **refinement of the requirement:** Defines **"how" it shall be realised**

| What? |
| --- |
| • R-12: The navigation system shall allow the driver to enter the destination of the trip conveniently. |

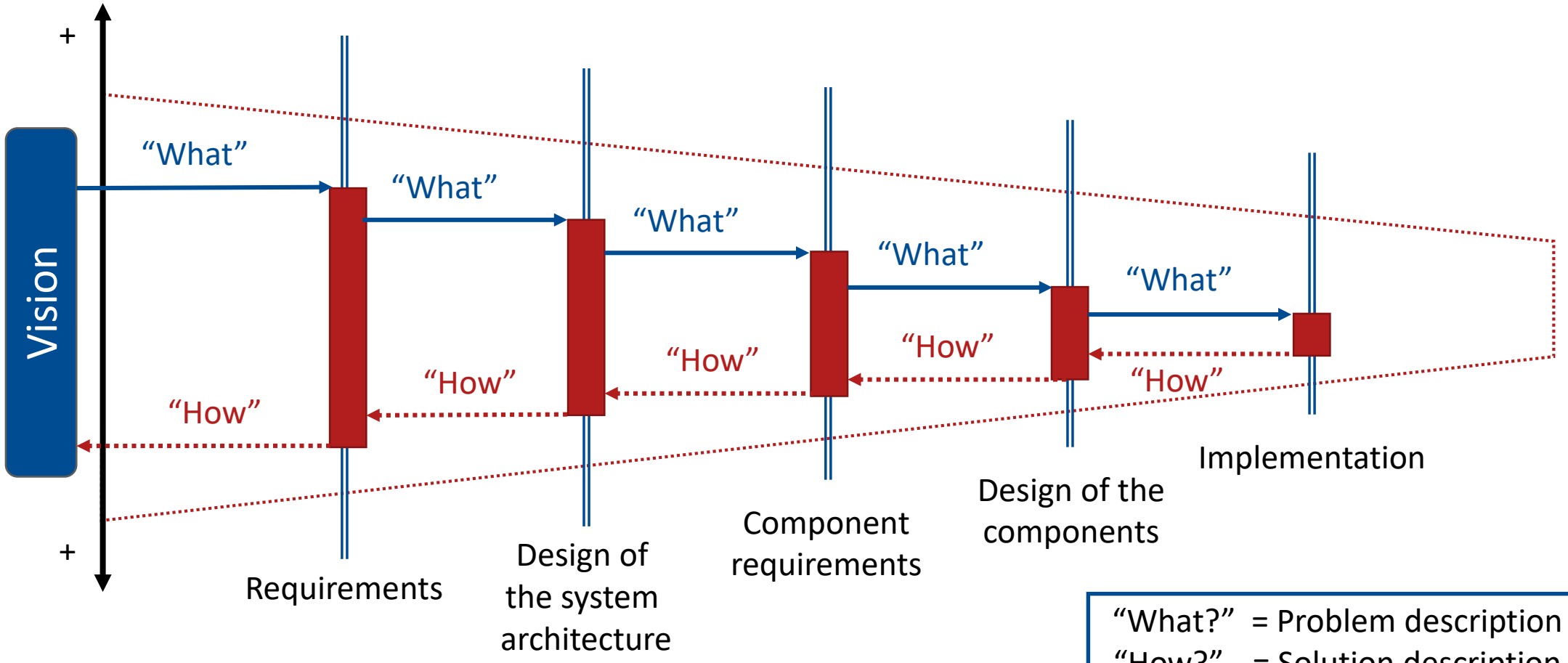
**Refinement of Requirements**

| How? |
| --- |
| • R-12.1: When the driver starts a new trip, the navigation system shall display a roadmap of the area centred on the current position.<br>• R-12.2: The navigation system shall allow the driver to scroll and zoom into the roadmap.<br>• R-12.3: After the driver has selected a destination on the roadmap, the system shall allow the driver to edit the destination details (city, street and house number). |

© SSE, Prof. Dr. Klaus Pohl

# In the System Development Process





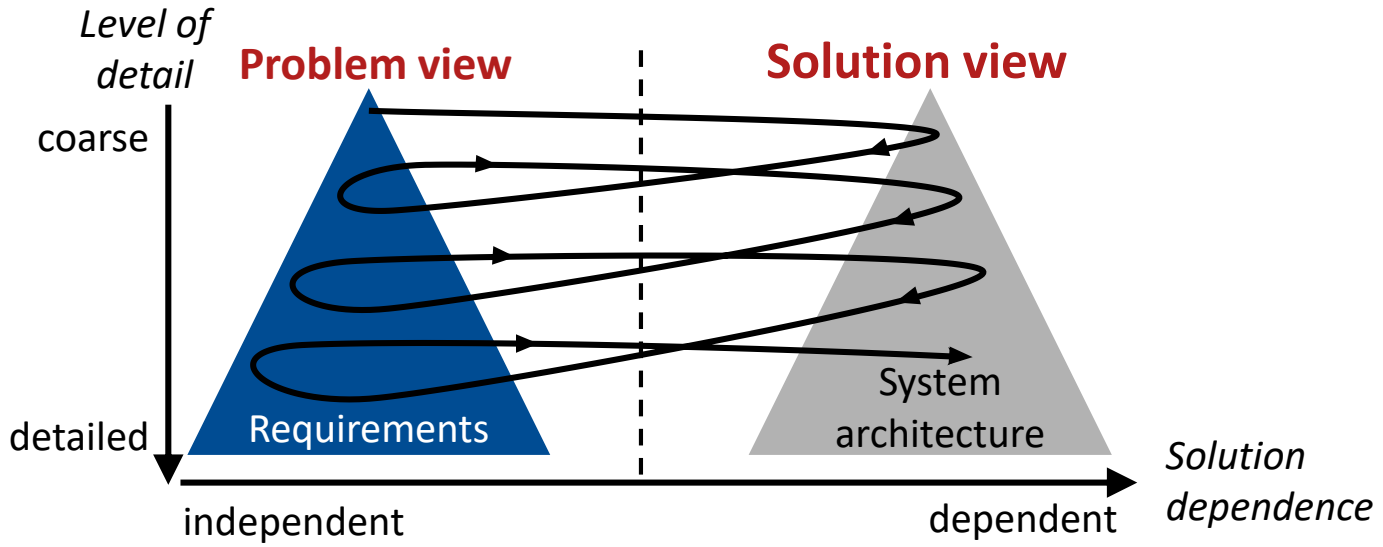

"What?" = Problem description
"How?" = Solution description

© SSE, Prof. Dr. Klaus Pohl

# Co-Evolution of Requirements and other Design Artefacts



- **Requirements define the problem** ("what should be developed")
- **System design specifies the solution** ("how the system should be developed")
- Requirements are the **basis for defining the architecture**, but architectural decisions also **influence the requirements**.
- **Interactions** between requirements and architecture:




based on [Ward and Mellor 1985, Nuseibeh 2001]

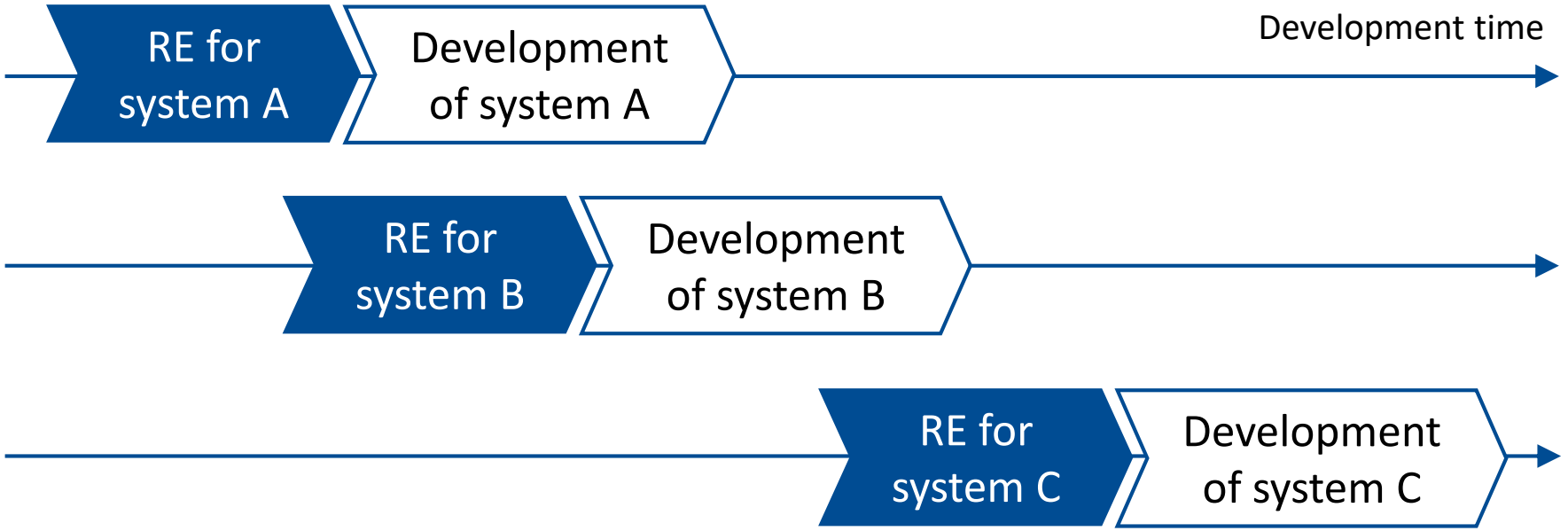
© SSE, Prof. Dr. Klaus Pohl

# Requirements Engineering as an Early Development Phase

© SSE, Prof. Dr. Klaus Pohl

# Requirements Engineering in Traditional Lifecycle and Process Models

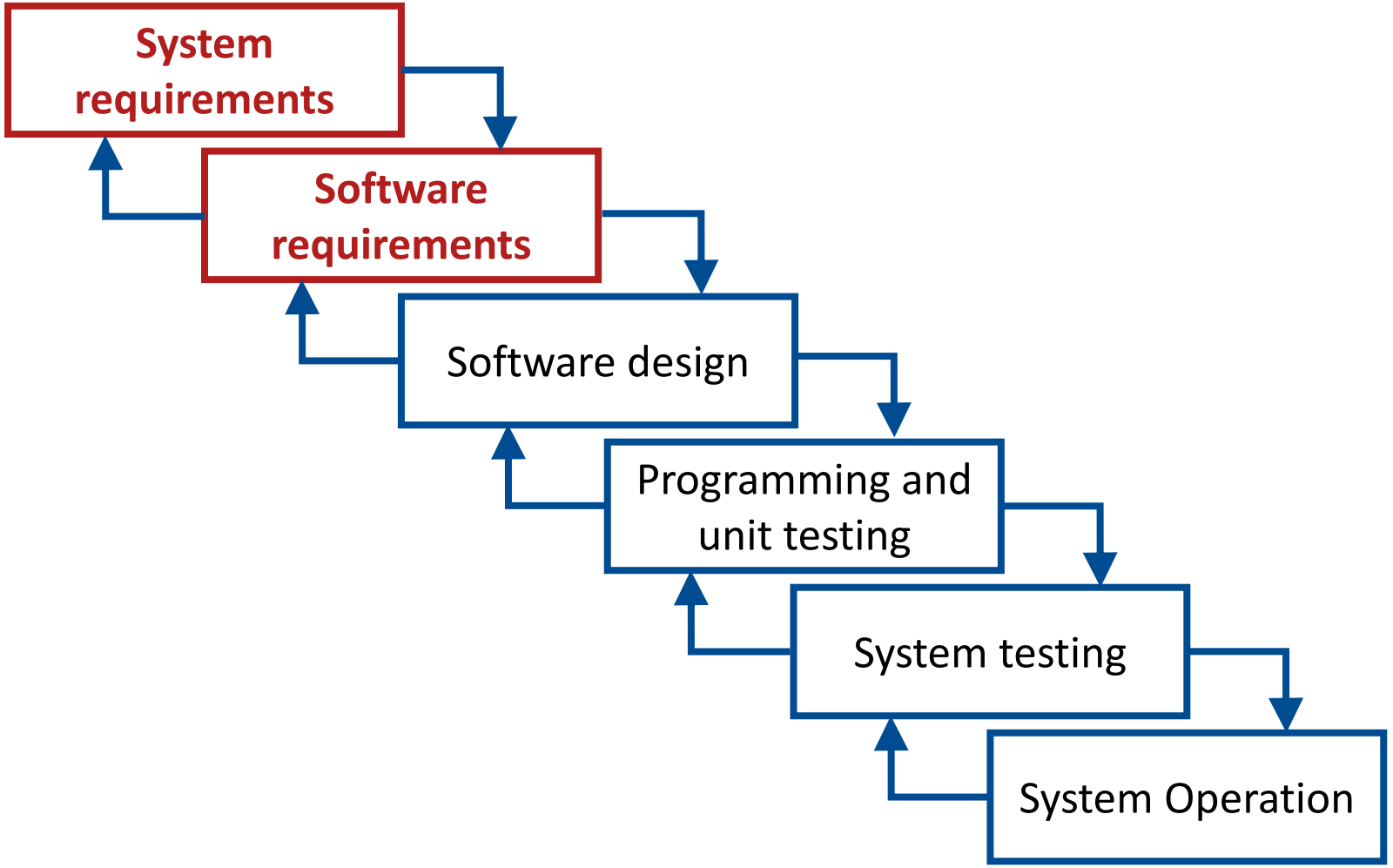
Requirements engineering (RE) is regarded as an **early (the first) phase of system development**.




© SSE, Prof. Dr. Klaus Pohl

# Development Process Models – Waterfall Model



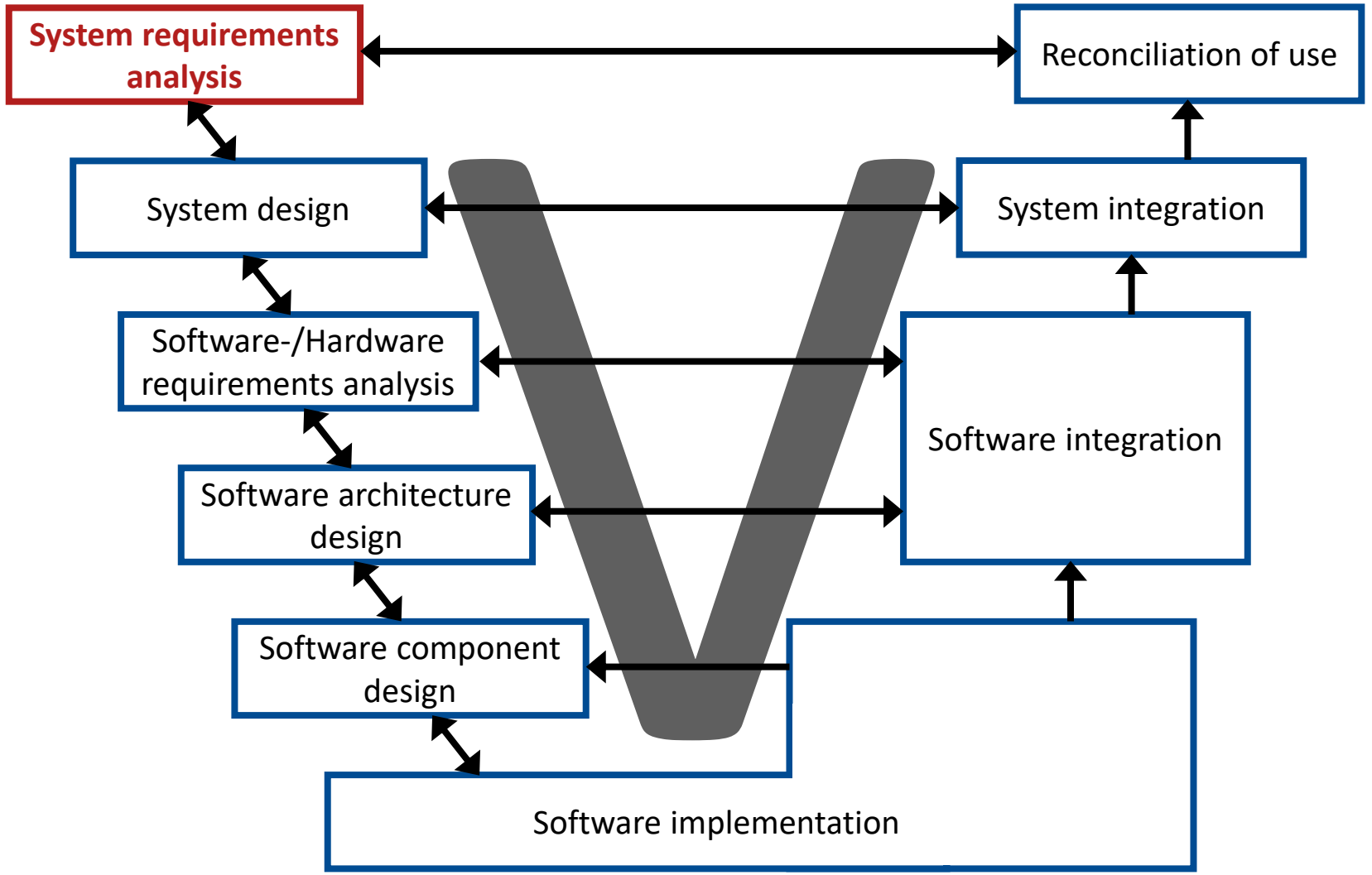



[Royce 1987]

© SSE, Prof. Dr. Klaus Pohl

# Development Process Models – V-Model





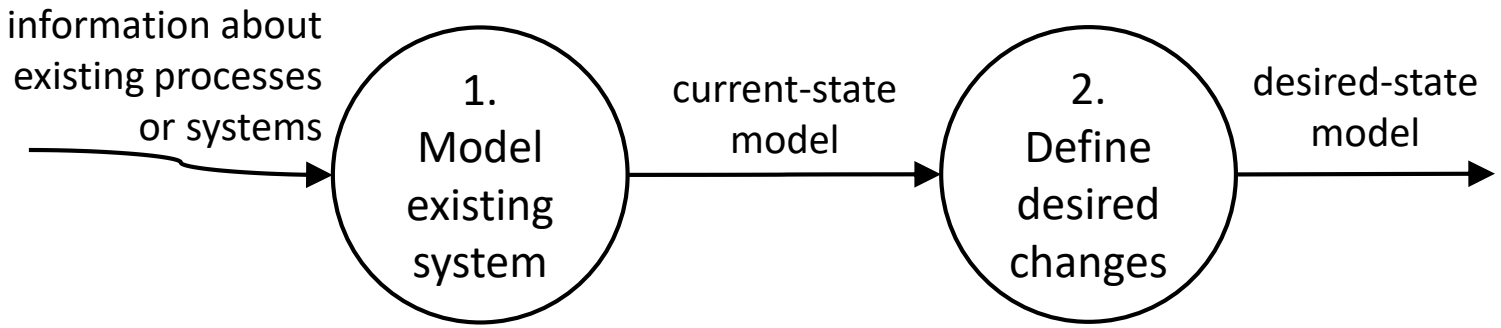

Based on [Dröschel and Wiemers 1999]

© SSE, Prof. Dr. Klaus Pohl

# 80's – early 90's: Traditional System Analysis



- Systems analysis aims at **understanding** and **defining requirements** of **existing** systems or processes in terms of function, data and behaviour.
- Typically the new system (partly) **automates** and thereby (partly) **replaces** existing systems and processes.

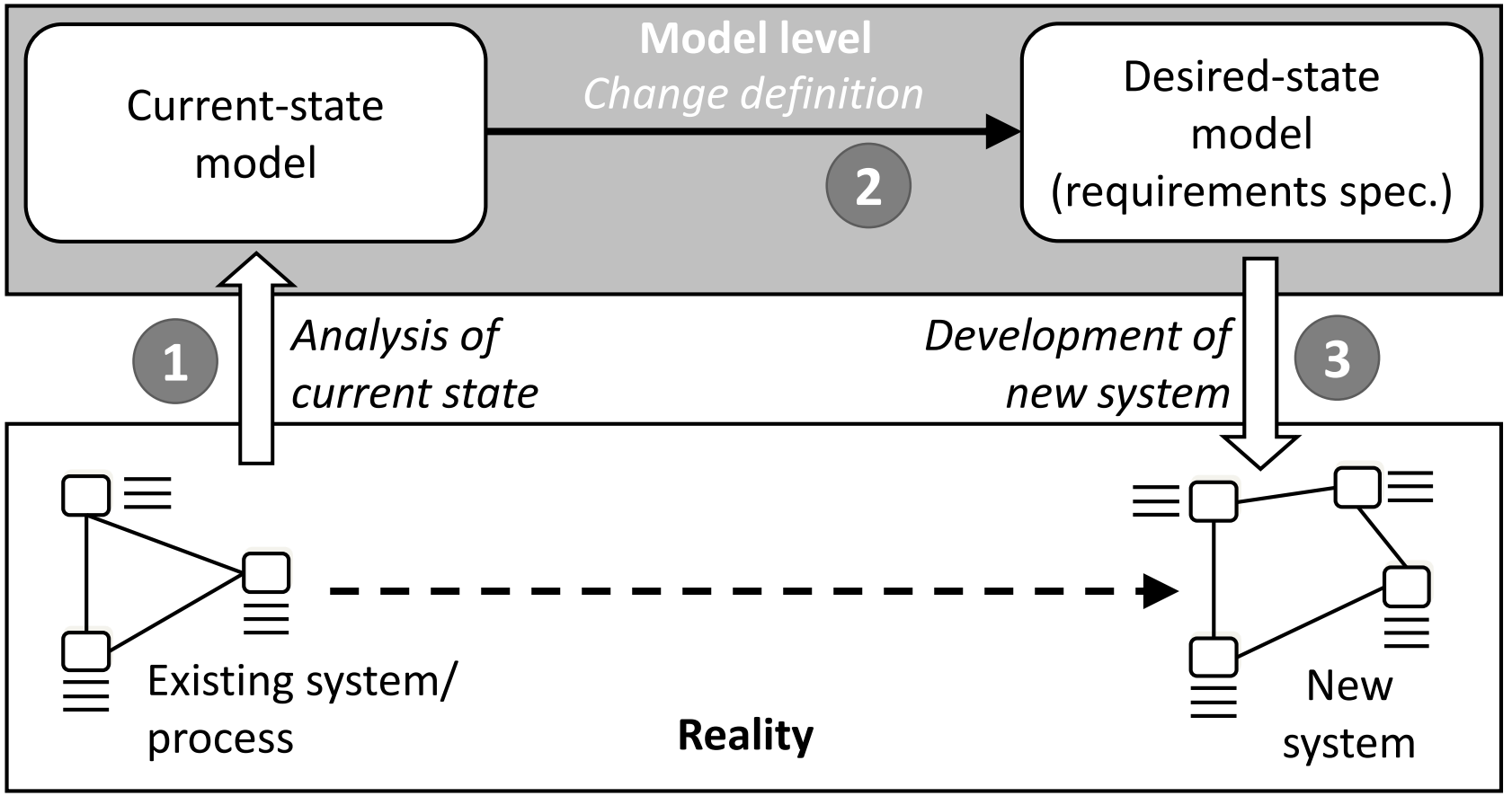



- Well known system analysis approaches are
  - **Structured Analysis** by DeMarco.
  - **Essential System Analysis** by McMenamin and Palmer [McMenamin and Palmer 1984]

© SSE, Prof. Dr. Klaus Pohl

# Current State and Desired-State Model






© SSE, Prof. Dr. Klaus Pohl

# Shortcomings (1)



- **No support for coping with requirements changes**
  - Requirements engineering is only performed as an early phase (limited time span).
  - Requirements are often inherently volatile!
  - Only suitable for projects in which the requirements are already well-understood at the beginning of the development process
- **No systematic sharing of knowledge**
  - No particular support for utilizing insights gained other development phases, which are relevant for the requirements.
- **Out-dated requirements**
  - Consequence: Requirements changes occurring later, e.g. during development, are not reflected in the requirements

© SSE, Prof. Dr. Klaus Pohl

# Shortcomings (2)



- **No systematic requirements reuse**
  - Reuse of requirements across project and product boundaries is not systematically supported.
  - Requirements have to be developed from scratch, even if a large set of requirements are relevant for more than one product/project.
  - Reuse happens, if at all, in an ad hoc manner.
- **Narrow focus**
  - Focus of requirements engineering: Typically restricted to the system under development
  - Important opportunities for product and process innovation are missed.

© SSE, Prof. Dr. Klaus Pohl

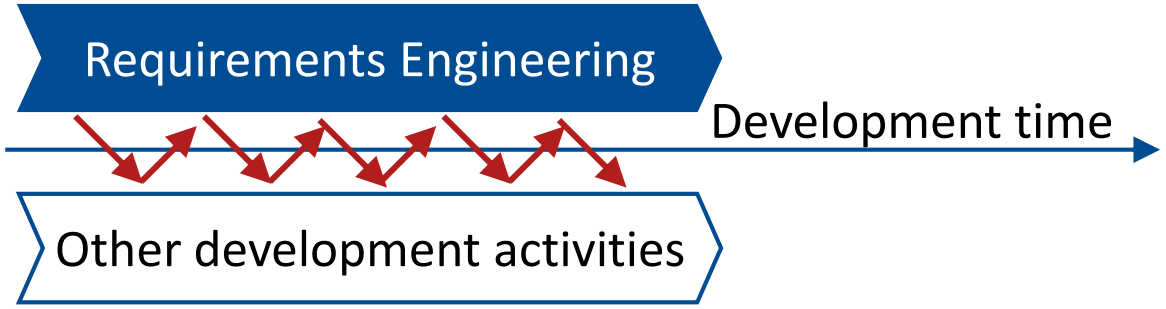
# Requirements Engineering as a Cross-Cutting Activity

© SSE, Prof. Dr. Klaus Pohl

# Requirements Engineering as a Cross-Cutting Activity



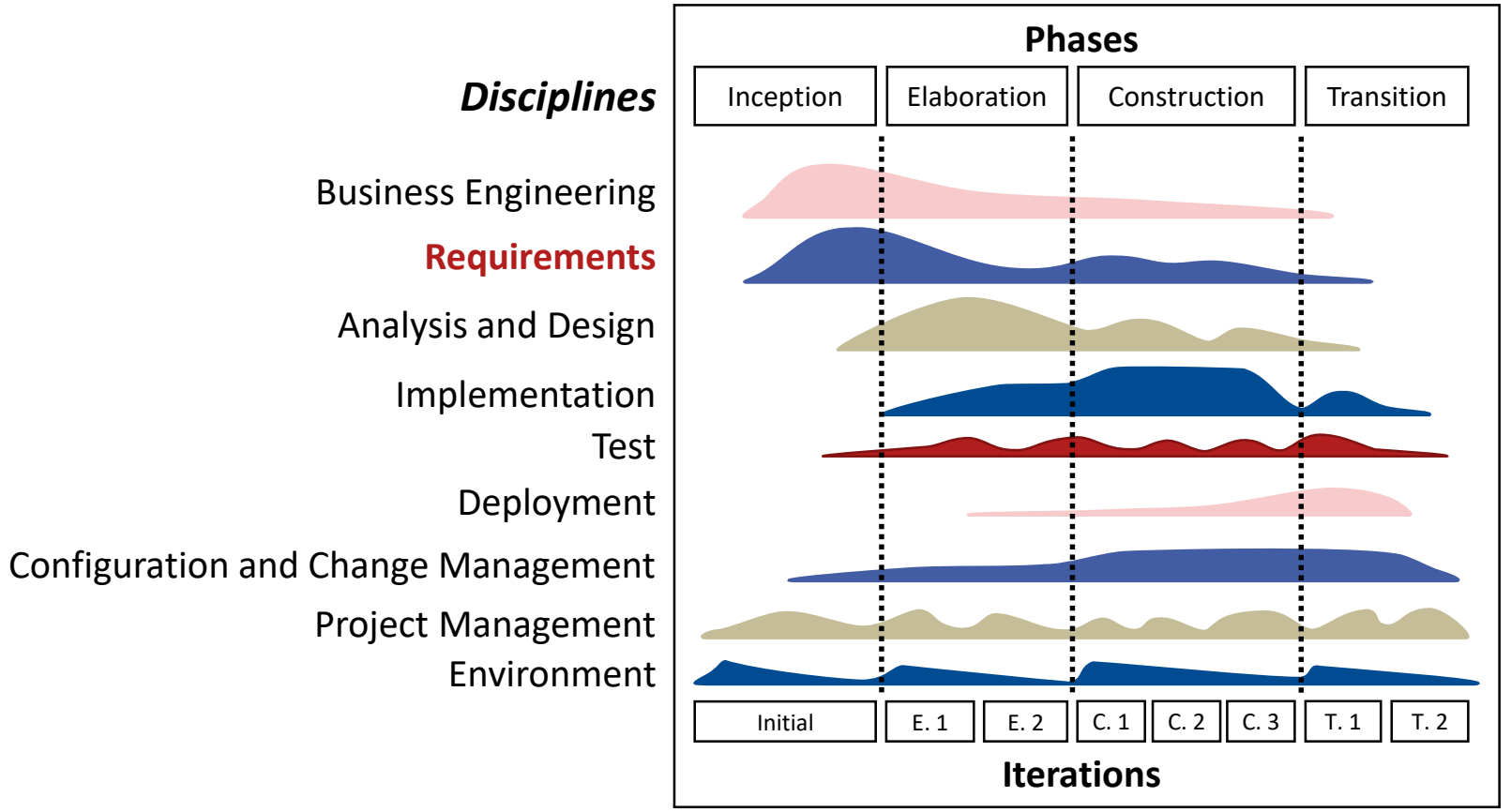
- **Requirements engineering** is a development activity **not limited** to the early phase(s) of a project.
- Requirements engineering **spans the entire development lifecycle**, i.e. across all project phases such as design, implementation or testing.




© SSE, Prof. Dr. Klaus Pohl

# Rational Unified Process

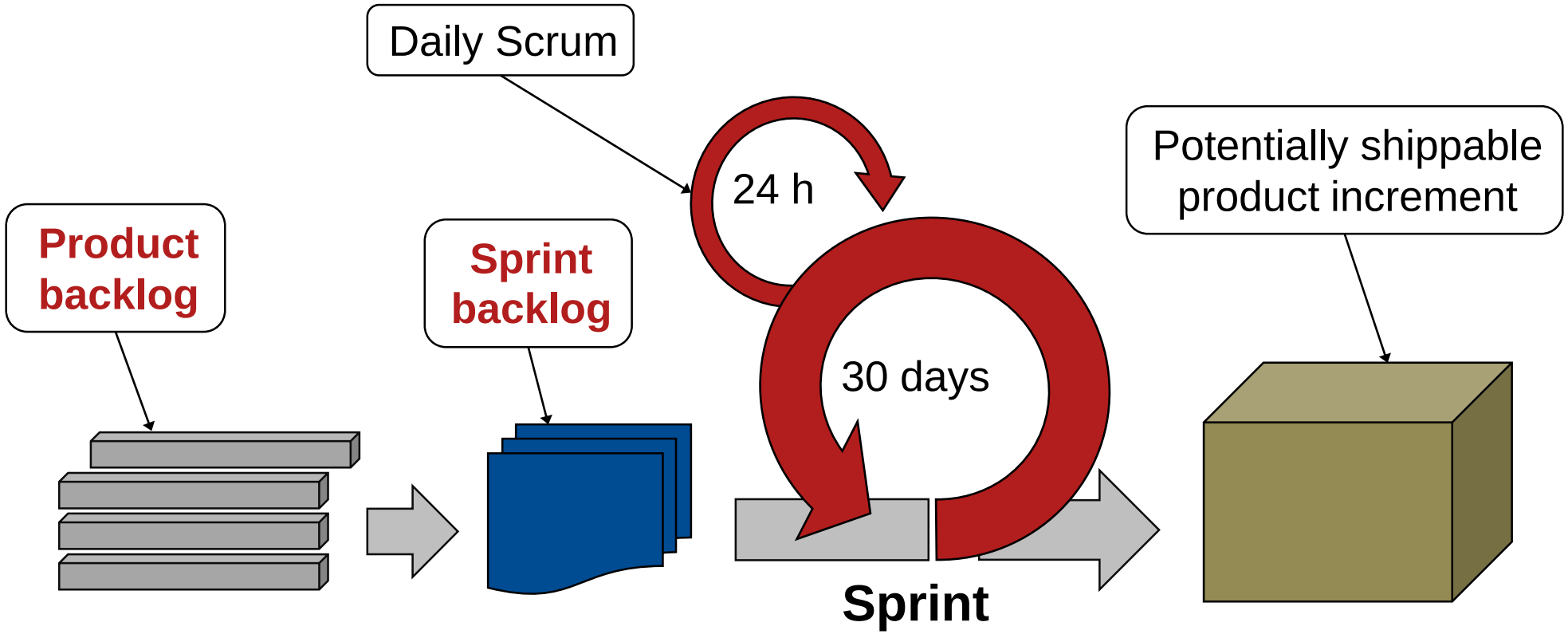





[Kruchten 2003]

© SSE, Prof. Dr. Klaus Pohl

# Scrum




[based on [Schwaber and Beedle 2001]

© SSE, Prof. Dr. Klaus Pohl

# Advantages



- **Managing requirements changes**
  - Addresses the need to cope with changes to inherently volatile requirements
  - Incremental delivery and customer interaction involves activities and systematic project management for:
    - Negotiating potential modifications of existing requirements artefacts
    - Integrating new ones at any time during the process
- **Up-to-dateness of requirements throughout the entire project**
  - Necessary modifications of requirements artefacts occurring in later phases can also be reflected in the original requirements artefacts.

© SSE, Prof. Dr. Klaus Pohl

# Remaining Shortcomings



- **Still no systematic reuse**
  - Reuse still limited to a specific project (e.g., for new requirements)
  - No reuse of similar requirements across projects and products
- **Still no systematic sharing of knowledge**
  - No particular support for utilising insights and gained in the course of one project in different projects
- **Focus still too narrow**
  - Innovation potential stemming from taking a cross-project and cross-product view on requirements may
- **No cross-project management of requirements and references**
  - Requirements only up-to-date within one project
  - No cross-references to requirements from other projects considered

© SSE, Prof. Dr. Klaus Pohl

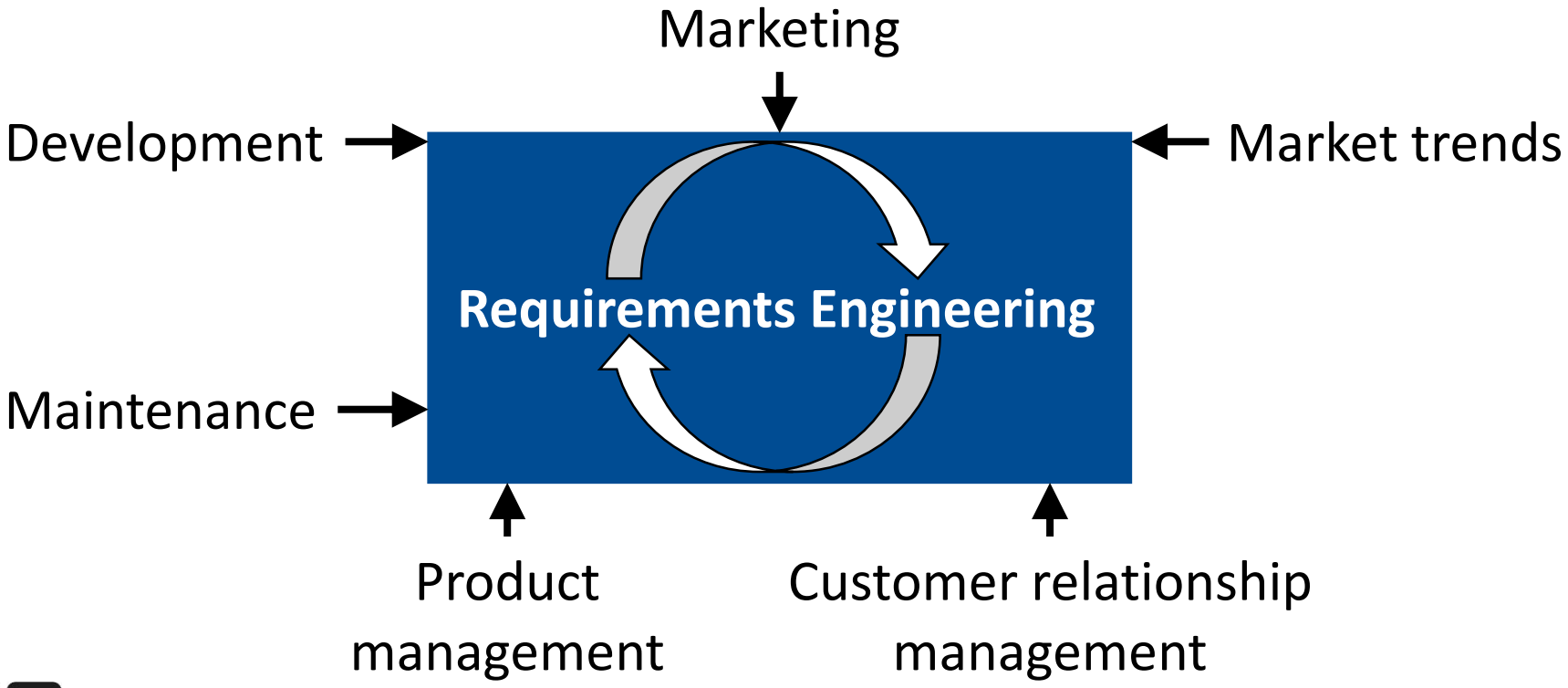
# Requirements Engineering as a Continuous Process

© SSE, Prof. Dr. Klaus Pohl

# RE as a Cross-Project and Cross-Product Activity (1)



- RE is implemented within an organization as **continuous activity across project** and **product boundaries**.
- The **inputs** from various sources are, e.g.:

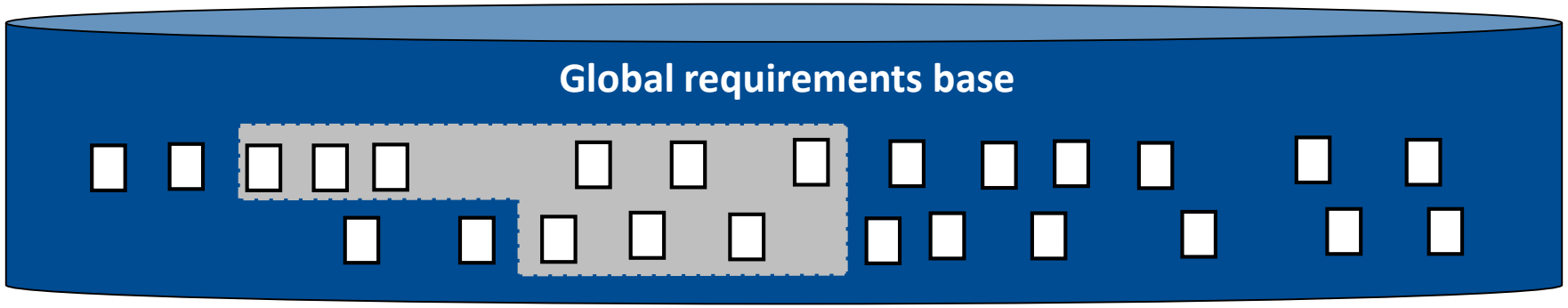



© SSE, Prof. Dr. Klaus Pohl

# RE as a Cross-Project and Cross-Product Activity (2)

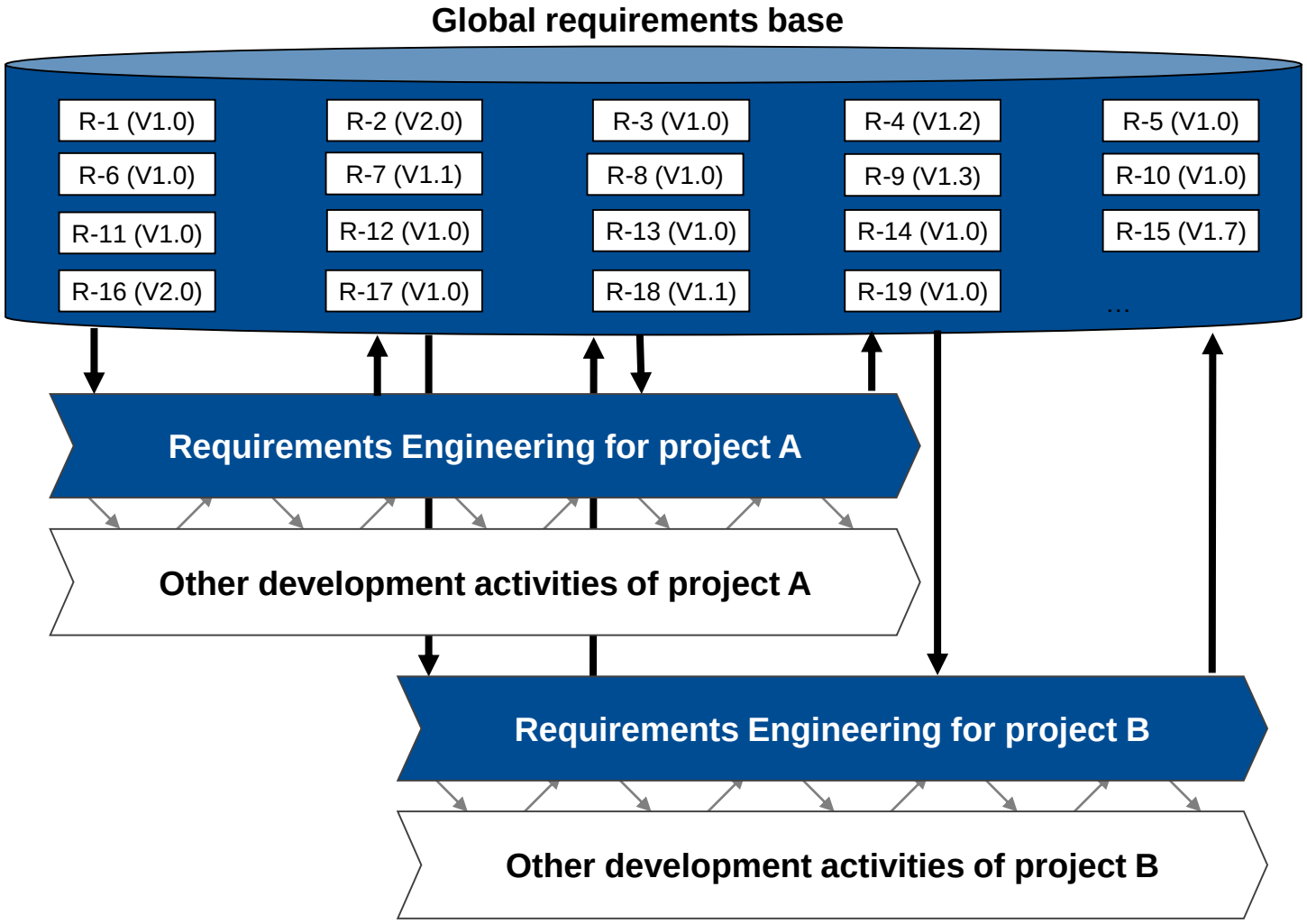


- **Continuously** create, change and delete requirements in a **global requirements base**.
- The **requirements base** contains requirements under development as well as agreed, complete and correctly specified requirements.
- At a given point in time a **set of requirements** to be **realized** in the next system or system release is selected from the requirements base.




© SSE, Prof. Dr. Klaus Pohl

# RE as a Cross-Project and Cross-Product Activity (3)




© SSE, Prof. Dr. Klaus Pohl

# Advantages (1)



- **Systematic learning and knowledge sharing**
  - Institutionalized learning process - involved stakeholders continuously exchange knowledge and experience
  - Clear responsibilities for the management of requirements across projects and products
  - Stakeholders continuously extend and improve their understanding of the requirements
- **Cross-project up-to-dateness of requirements**
  - Up-to-date requirements base across different products and projects
  - Ensures the integration of changes into the requirements base
  - Requirements references across different products are also up-to-date

© SSE, Prof. Dr. Klaus Pohl

# Advantages (2)



- **Systematic reuse across projects and products**
  - Facilitates the reuse of requirements and related development artefacts across different projects and products
- **Broad and unconstrained focus**
  - Fundament for taking full advantage of innovation potential stemming from cross-project and cross-product considerations
  - Identification of system analogies
  - Inspiration from new technologies used in other projects

© SSE, Prof. Dr. Klaus Pohl

# Summary



- Non-functional requirements are underspecified functional requirements and/or quality requirements!
- Requirements engineering is embedded within organizational processes.
- Traditional system analysis is typically performed as an early phase of system development.
- Requirements engineering as an early phase of system development has significant shortcomings.
- Requirements engineering as a cross-cutting development activity has advantages, but still some shortcomings.
- Requirements engineering as a continuous process spanning independent of project and product boundaries overcomes these shortcomings.

© SSE, Prof. Dr. Klaus Pohl

# Literature (1)


[DeMarco 1978] T. DeMarco: Structured Analysis and System Specification. Prentice Hall, Englewoods Cliffs, New Jersey, 1978.

[Dröschel and Wiemers 1999] W. Dröschel, M. Wiemers: Das V-Modell 97. Der Standard für die Entwicklung von IT-Systemen mit Anleitung für den Praxiseinsatz. Oldenbourg, 1999.

[Haumer et al. 1998] P. Haumer, K. Pohl, K. Weidenhaupt: Requirements Elicitation and Validation with Real World Scenes. IEEE Transactions on Software Engineering, Vol. 24, No. 12, 1998, pp. 1036-1054.

[Kruchten 2003] P. Kruchten: Rational Unified Process: An Introduction (Addison-Wesley Object Technology Series). 3rd ed., Pearson Addison Wesley Prof, 2003.

[McMenamin and Palmer 1984] S. M. McMenamin, J. F. Palmer: Essential Systems Analysis. Prentice Hall, London, 1984.


© SSE, Prof. Dr. Klaus Pohl

# Literature (2)



[Nuseibeh 2001] B. Nuseibeh: Weaving Together Requirements and Architectures. IEEE Computer, Vol. 34, No. 3, IEEE Computer Society, Los Alamitos, 2001, pp. 115-117.

[Royce 1987] W. W. Royce: “Managing the development of large software systems”, IEEE WESCON, 1970, pp. 1-9.

[Schwaber and Beedle 2001] K. Schwaber, M. Beedle: Agile Software Development with Scrum. Prentice Hall, Upper Saddle River, 2001.

[Ward and Mellor 1985] Structured Development for Real-Time Systems: Introduction and Tools. Prentice Hall, Upper Saddle River, N.J., 1985.

© SSE, Prof. Dr. Klaus Pohl

# Image References


[1] Licensed by http://www.iconshock.com/

[2] Provided by Microsoft Office

[3] https://www.tizen.org/about/tizen-brand-guidelines

[4] https://ubports.com/de/design-guidelines


# Legend


Definition


Example

© SSE, Prof. Dr. Klaus Pohl

UNIVERSITÄT DUISBURG ESSEN

*Offen im Denken*

# Vielen Dank für Ihre Aufmerksamkeit

© SSE, Prof. Dr. Klaus Pohl

PALUNO
The Ruhr Institute for Software Technology