An Approach for Requirements Engineering for Software Library-Components and Patterns to be Reused in and across Product Lines

Dr. Jens Liebehenschel

metio
methodik · information · organisation

Dirk Herrmann

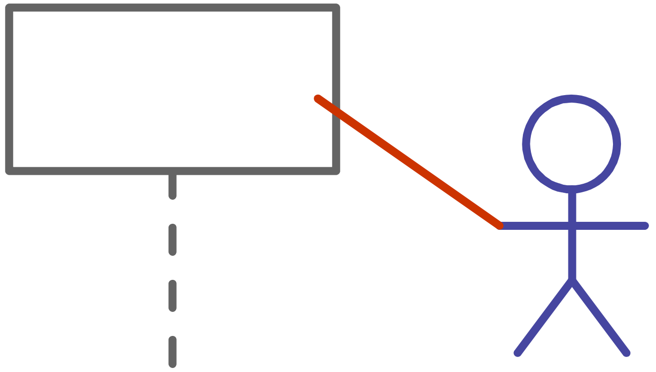
BOSCH

About metio

methodik · information · organisation

- System- and Software-Engineering
  - Architecture development and analysis
- Consulting
  - Method development
    - Architecture and product line approaches
    - Scoping
- Training

square-m



Contents

Classification

- Method developed and applied in practice
  - Requirements engineering
  - Requirements documentation

- Development
  - *Assets* to be used in different contexts
    - Software components
    - Patterns

Motivation

- No lightweight approach for given situation available
    - Method
    - Tools
    - Templates

Our Situation

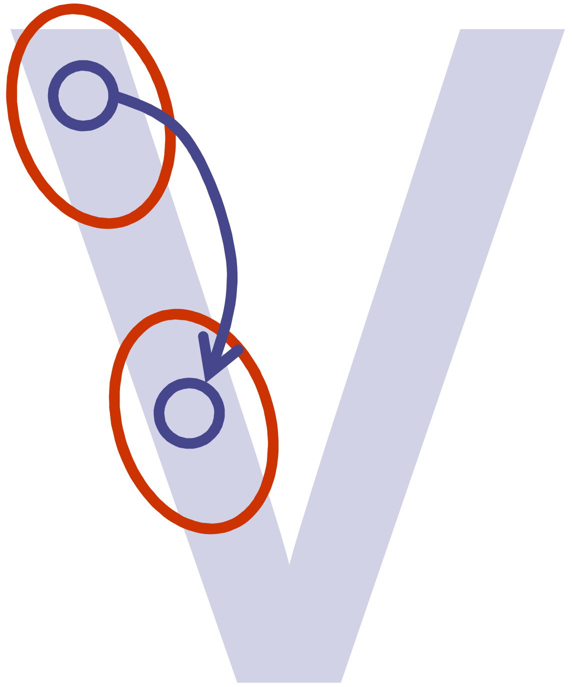
- Assets to be developed and substituted
  - Reuse across product lines necessary
  - Asset scope roughly known
  - Detailed asset requirements not known
  - No standardized interfaces
  - Not necessarily one solution for all projects

One Project

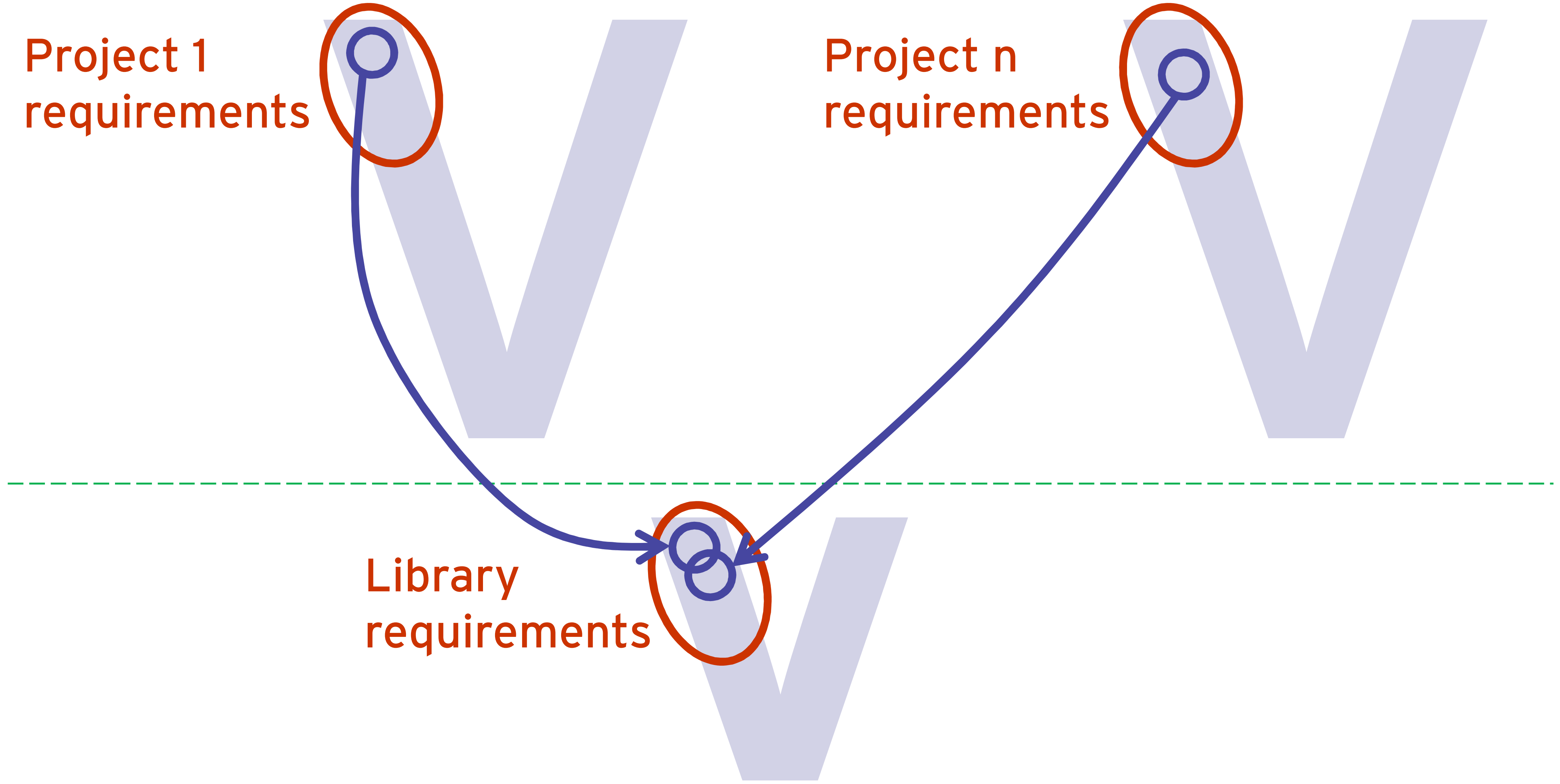
Break-down project requirements to library requirements during development

Project requirements

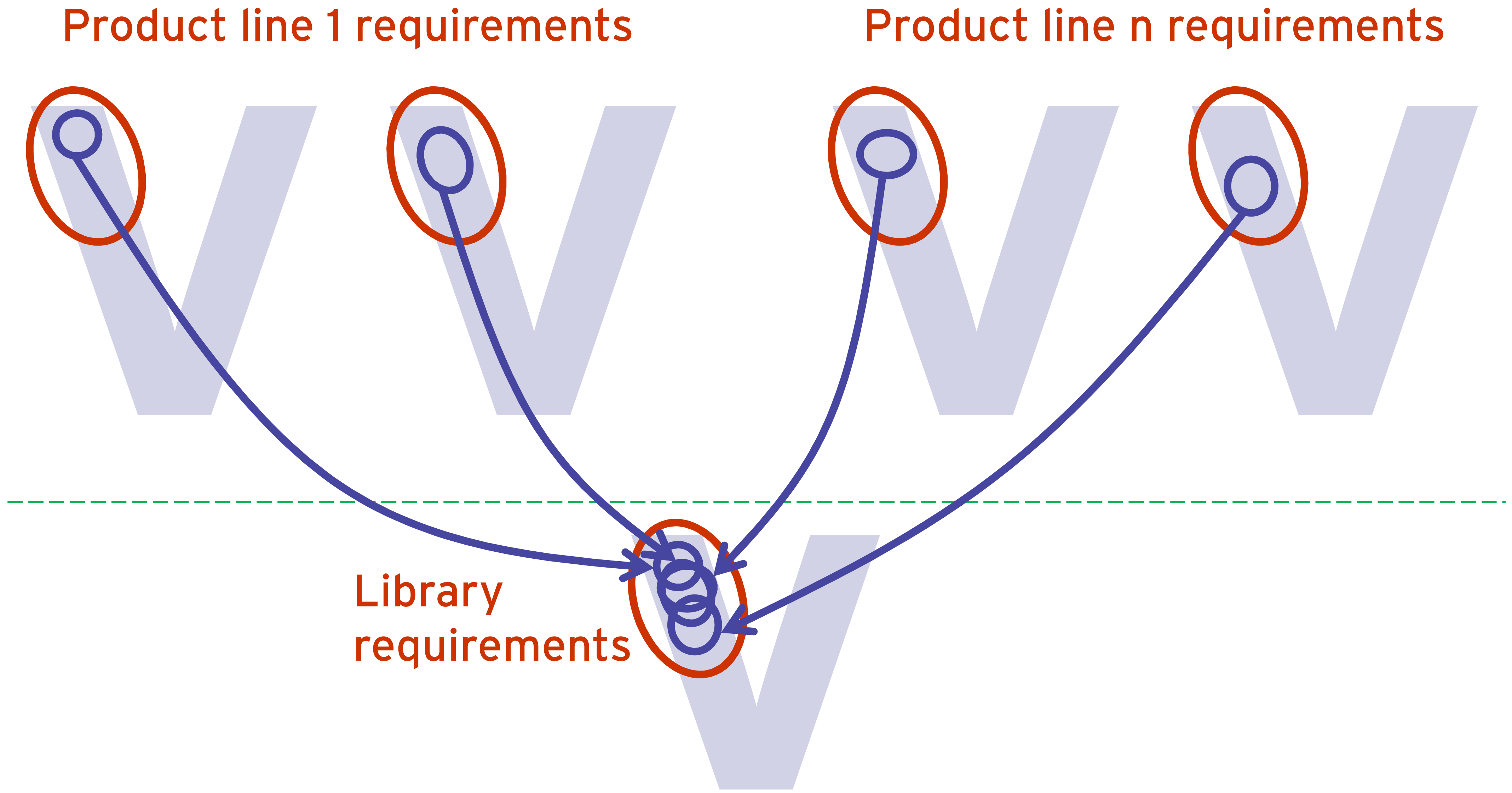
Library requirements



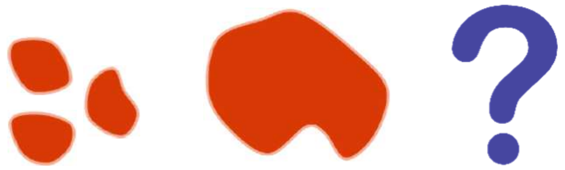
Several Projects



Our Situation – Several Product Lines



Challenges



- Technical
  - Asset scope not completely defined
  - Interfaces are subject to design decisions
  - Variability in the product lines
- Non-technical
  - Economical optimum under consideration of
    - Development and test
    - Configuration and integration

Possible Approach

- ~~Synchronize product lines' requirements~~
  - ~~Variability~~
  - ~~Internal interfaces~~
  - ~~Behavior of internal components~~

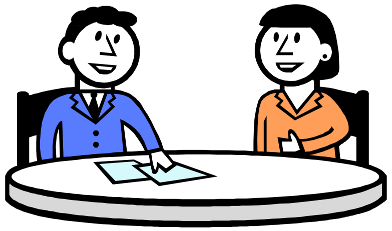
Very difficult!

Our Approach

- Identification of the variability
- Definition of the asset scope
  - Agreement on useful set of requirements
  - Structure and functionality that fits to most product lines / projects

Keep it simple!

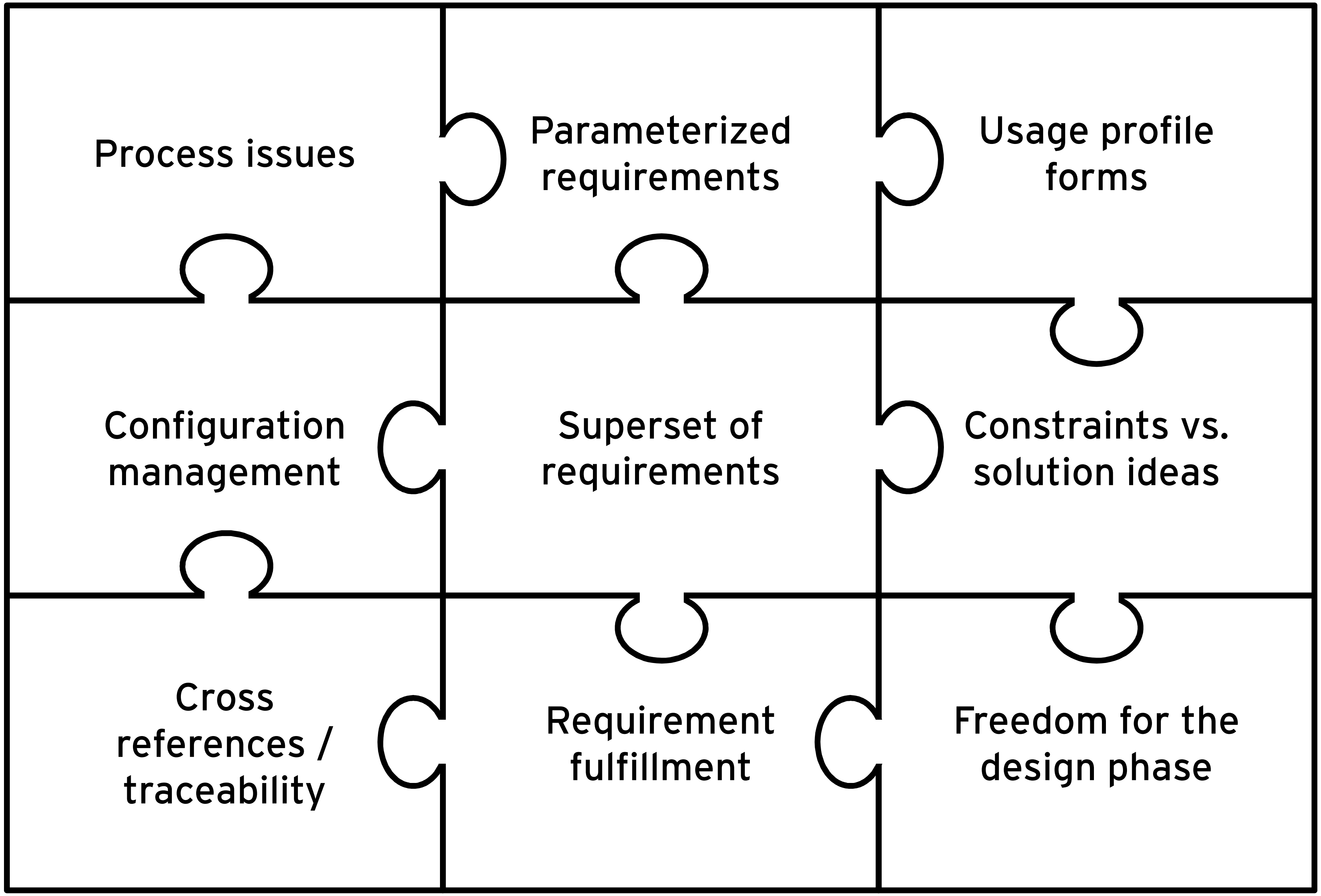
Approach – Interviews



- Interviews with stakeholders
  - Documentation grows

- Rules for interviews
  - Elicit real requirements – Why? Why? Why?
  - Negotiation
    - Important features
    - Prioritization

Approach – Documentation



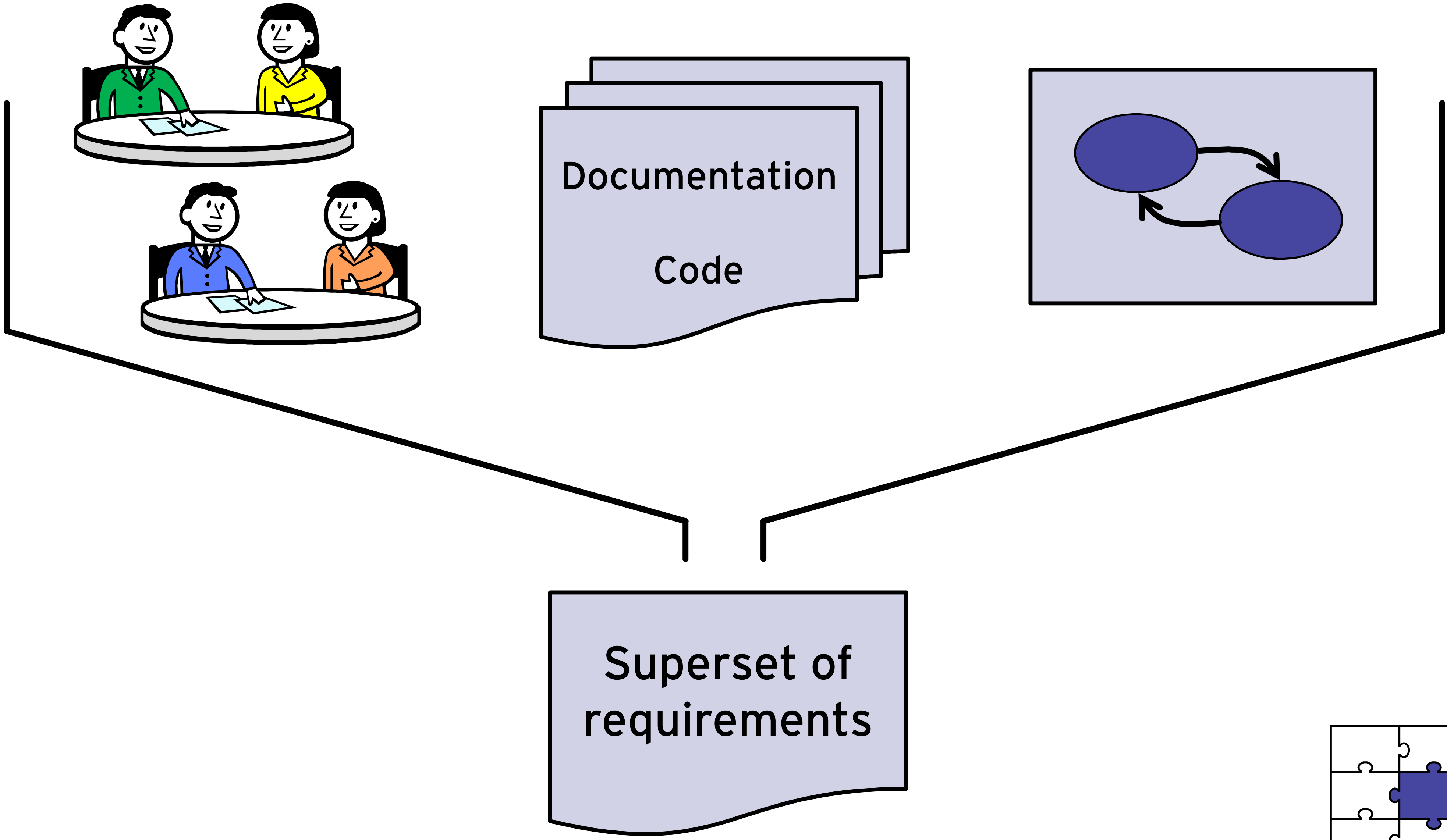
Process issues
Parameterized requirements
Usage profile forms
Configuration management
Superset of requirements
Constraints vs. solution ideas
Cross references / traceability
Requirement fulfillment
Freedom for the design phase

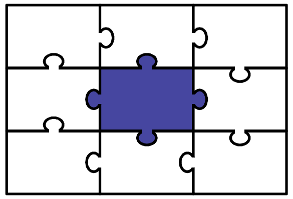
Superset of Requirements

Content

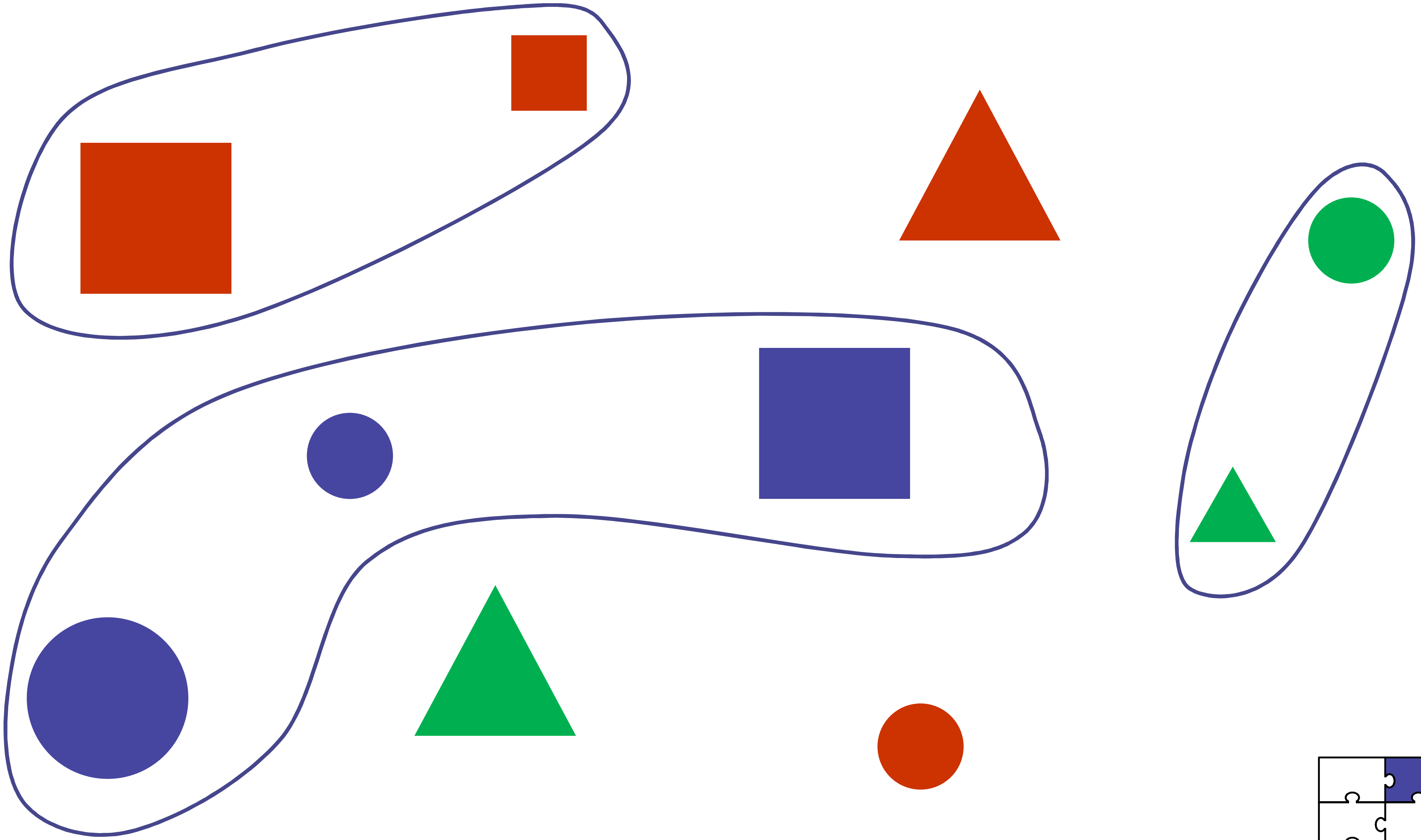
- Requirements from all potential projects
  - Requirements in the close environment of the asset
  - Contradictive requirements
  - Solutions
  - Constraints
  - Potential future requirements
  - Non-requirements
- Assignment to the projects

Superset of Requirements





Parameterized Requirements



Parameterized Requirements

- Criteria for abstraction
  - Common sense
  - Language and abstraction of projects

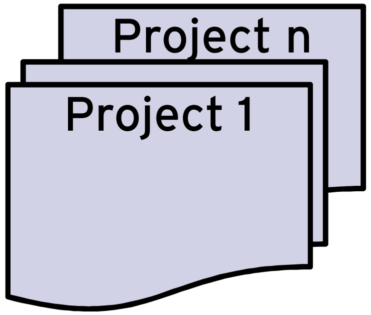
- Benefits
  - ⊕ Elimination of redundancy
  - ⊕ Fewer requirements
  - ⊕ Overview on variability

Parameterized Requirements

Requirement	Project A	Project B
The function shall be usable within {left hand drive cars, right hand drive cars, both}.	Both	Left hand drive cars
The driver's side shall be determined at the time of {build, system manufacturing, car manufacturing, system boot}.	System boot time	Build time
The response time of the function shall be below {time}.	below 500ms, but not smaller than 100ms	250ms

Asset Specific Usage Profile Forms

- Extension of the aspect "Parameterized Requirements"
- Usage profile forms
  - Filled out by each project
  - For fine grained information
- Example
  - Properties of data to be stored persistently

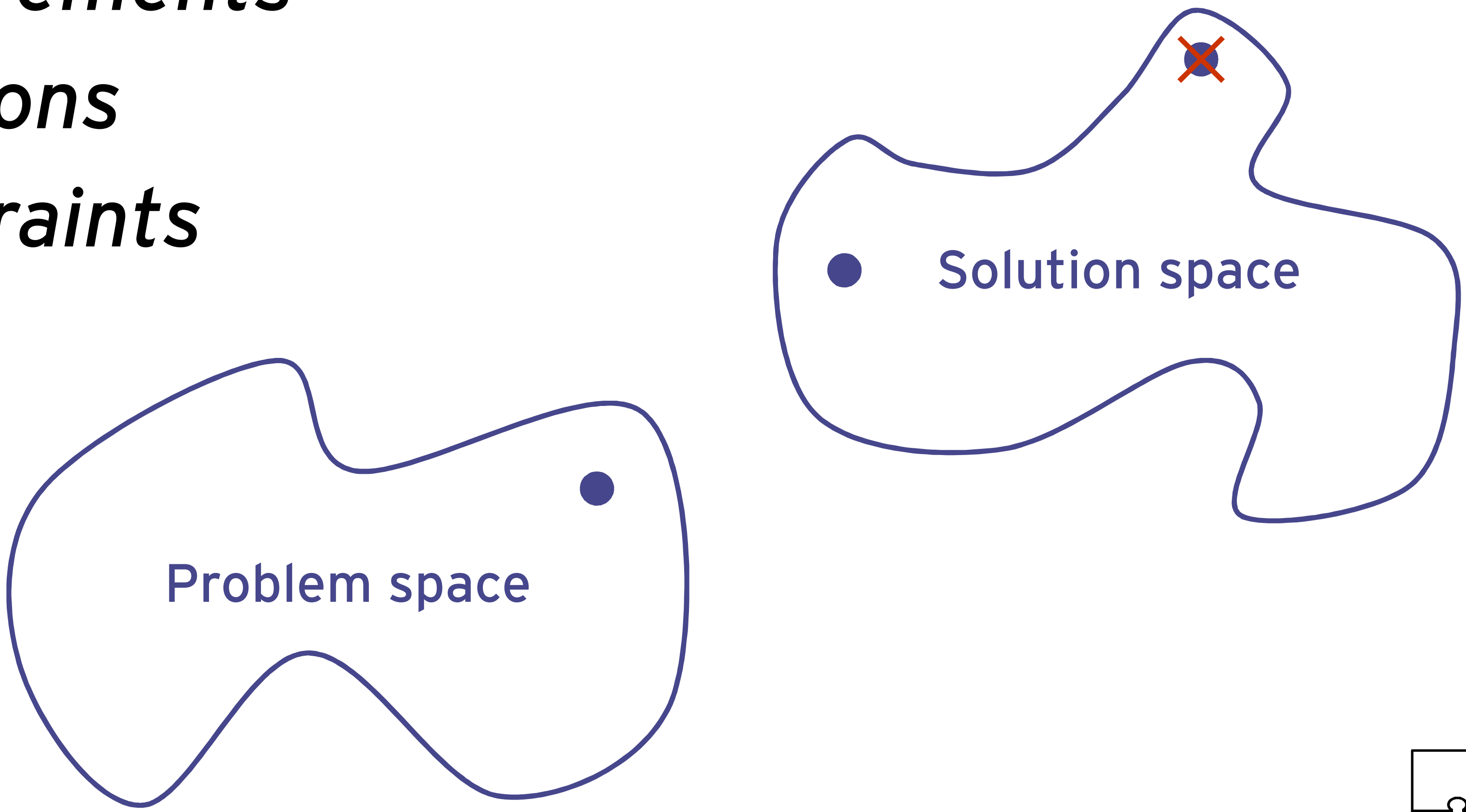


Asset Specific Usage Profile Forms

- Benefits
  - Good overview
  - Trend available, even if the information slightly changes during development

Constraints vs. Solution Ideas

- Classes of “requirements”
  - *Requirements*
  - *Solutions*
  - *Constraints*



Constraints vs. Solution Ideas

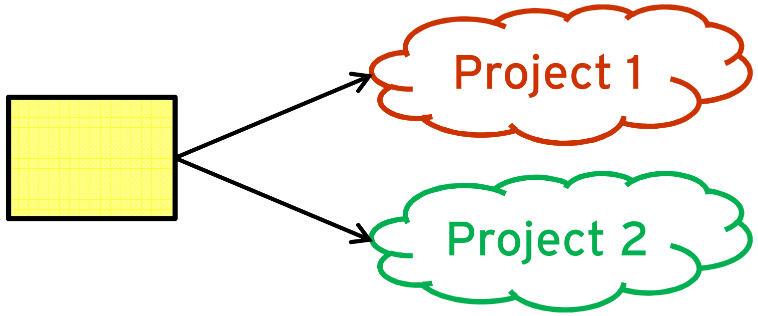
- Are solution concepts requirements?
  1. Underlying requirements difficult to express
     - Solutions were kept as ideas for the design
     - Real requirements were connected to the solutions
  2. Certain solutions are demanded / excluded
     - Solutions become constraints

Freedom for the Design Phase

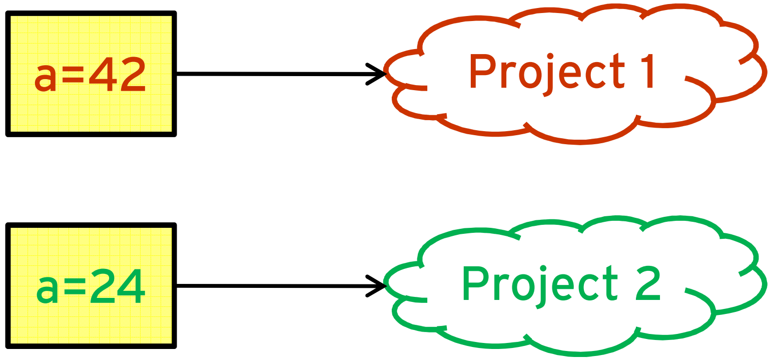
Requirement	Project A	Project B
The component shall be usable in an environment without preemptive scheduling.	Yes	No
If any input signal is unavailable, then the component shall not influence the actuators.	Yes	Yes, but also if signal x is unavailable
If any input signal is unavailable, then the component shall make this visible to the error memory unit.	Yes	

Freedom for the Design Phase

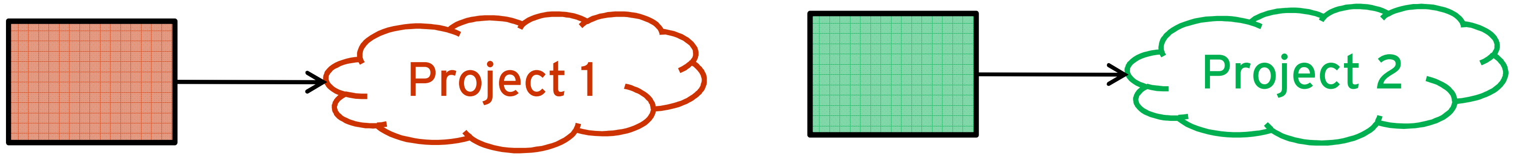
- Common library for different projects



- Configurable library

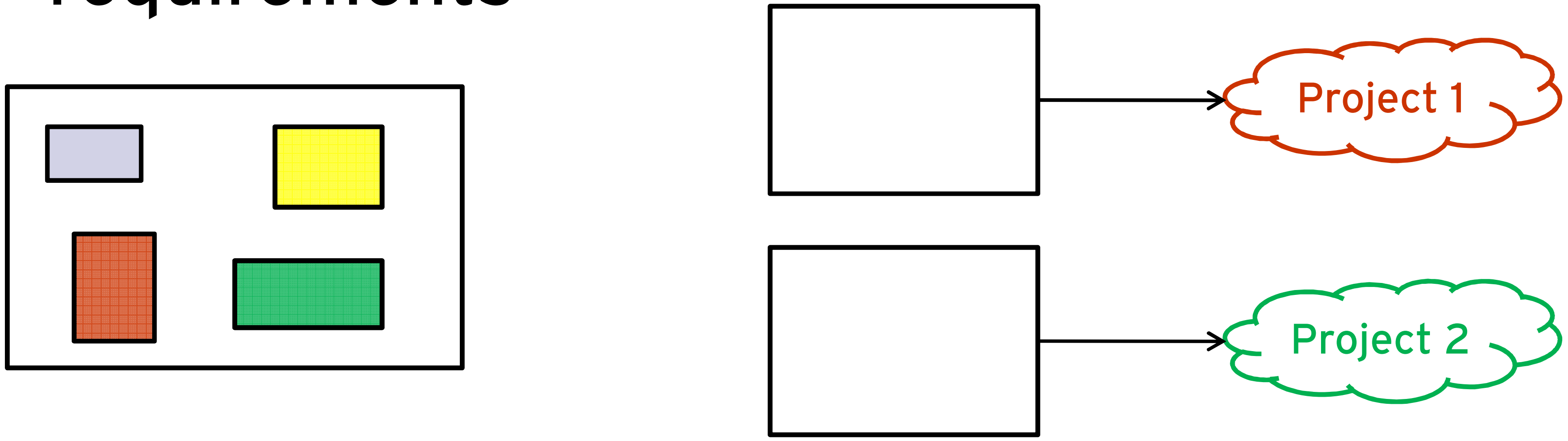


- Different libraries for different projects

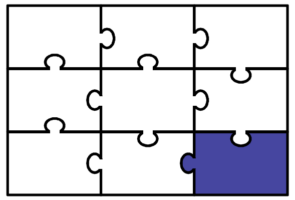


Freedom for the Design Phase

- Customizable set of library elements
  - Integration by the projects – based on requirements

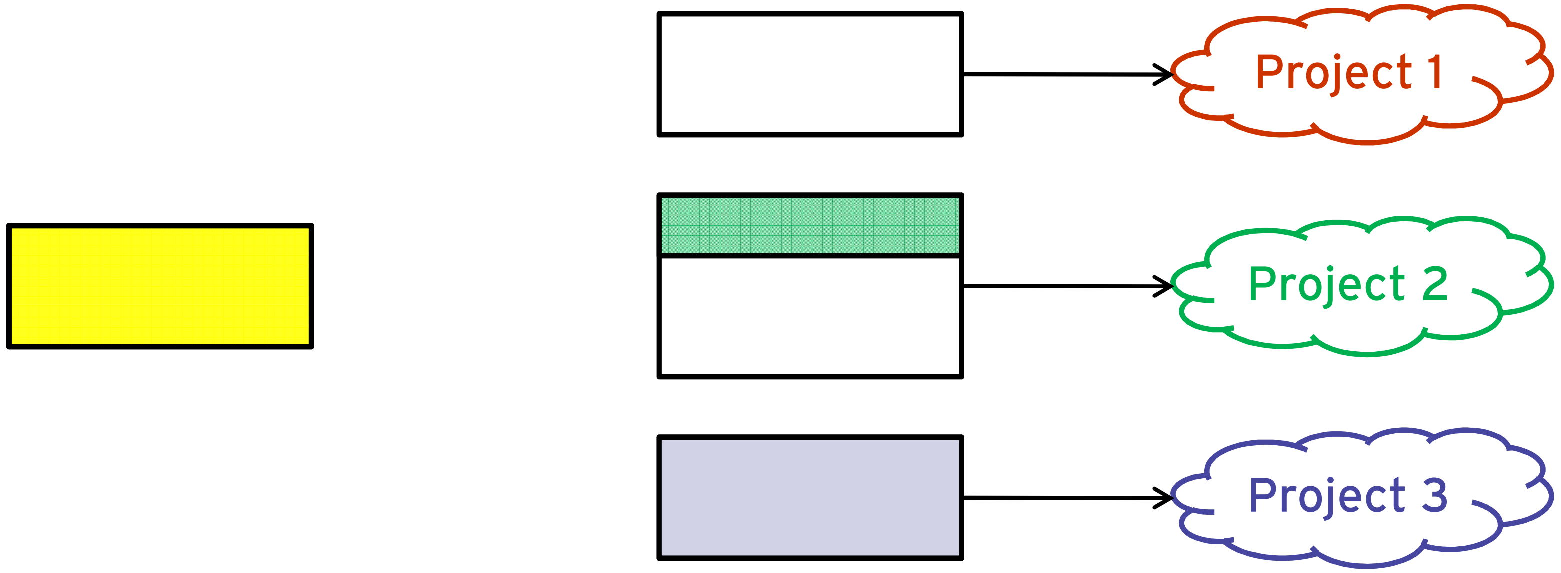


- Combination of the approaches



Fulfilling requirements vs. supporting fulfillment

- Not all requirements of all systems have to be fulfilled by the library
  - but they shall be satisfiable, e.g. by adding some additional functionality



Retrospect on the Methodology

- Agile approach
  - Method was developed and applied in parallel
  - Patterns were applied to design the approach
    - Separation of concerns
    - Abstraction
    - Keep it simple
  - Structure of the documentation was improved continuously
    - Requirements documents
    - Usage profile forms

Success Story

- Benefits of the approach
  - ⊕ Communication with stakeholders and designers
  - ⊕ Overview commonalities and variability
  - ⊕ No unnecessary restriction of the design space
  - ⊕ Transfer of know-how between projects
  - ⊕ Questionnaires for future projects
  - ⊕ Lightweight approach

Success Story

- Approach successfully applied to some
  - Library components
    - Development of new software component
    - Substitution of several implementations by one solution
    - Redesign of existing libraries
  - Patterns
    - Development of reusable patterns

Success Story

- One example: SW library
  - Shortly after design and implementation used in 17 projects belonging to 7 product lines
  - No problems in the field until now

Thank you!