Variability Management in the Industrial Automation Domain

Daniela Rabiser

CDL MEVSS, Johannes Kepler University Linz, Austria
daniela.rabiser@jku.at

Invited Talk KV PLE – May 18, 2016 – JKU Linz

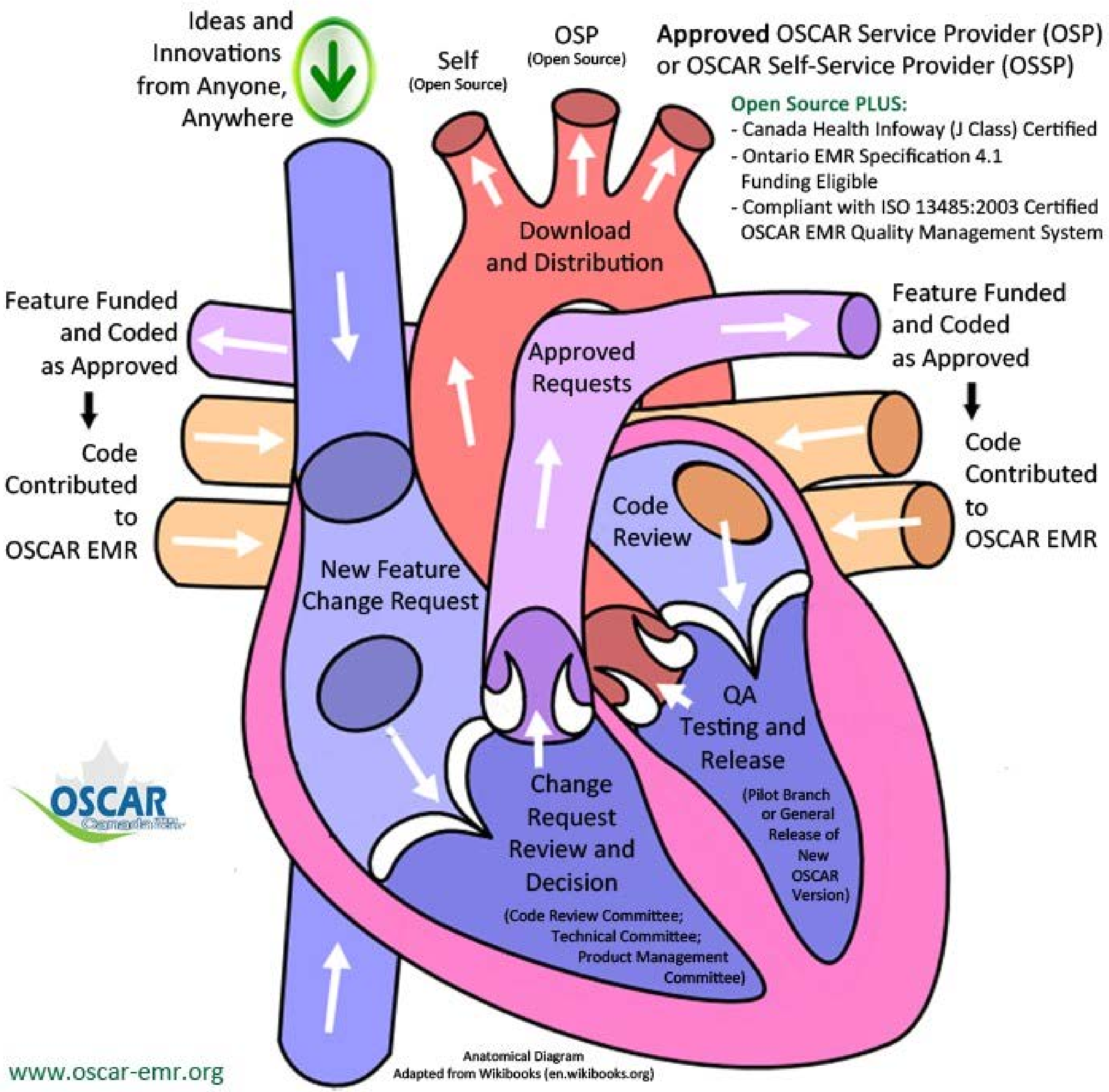
Research Context

- **Software ecosystems (SECOs)**
  - Involve multiple organizations and PLs
  - *“…set of businesses functioning as a unit and interacting with a shared market for software and services...“ [Jansen 2009]*
  - *“… a community of 3rd party application developers, around a successful product…” [Bosch 2010]*

- **Industrial software ecosystems (ISECOs)**
  - *“...mainly internal business units with different motivations and interests...” [Schultis 2013]*
  - Clone-and-own reuse as common practice for creating product variants [Rubin 2013]

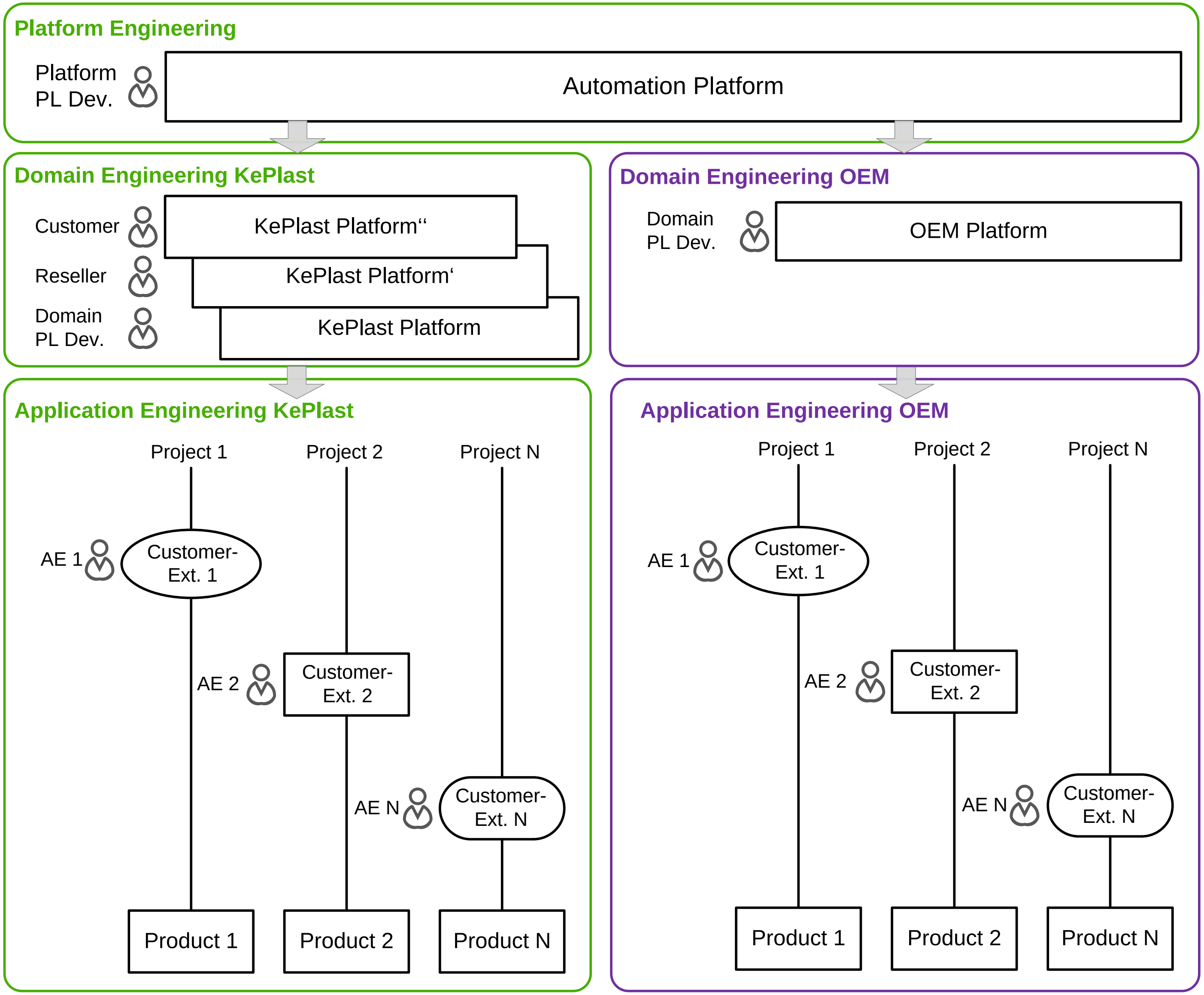
Software Ecosystem Examples (1/2)

Software Ecosystem Examples (2/2)



- Electronic Medical Record (EMR) system
- Supports over 1.5 million patients across Canada
- Developed by a community of universities, hospitals, service providers, developers, etc.

Industry Context: Keba‘s ISECO



Trends in Systems and Software Variability

[Bosch et al. 2015]

- Very successful and *almost* perfect product
- Customer-specific versions needed
- Required changes must be implemented for most product versions
  - Inefficient, time-consuming, and error-prone task

- Introducing a platform improves development efficiency
- However, new challenges arise and **variability management** is required

FOCUS: GUEST EDITORS' INTRODUCTION

TRENDS IN SYSTEMS AND SOFTWARE VARIABILITY

Jan Bosch, Chalmers University of Technology

Rafael Capilla, Rey Juan Carlos University

Rich Hilliard, consulting software systems architect

THE FOLLOWING STORY has happened thousands of times. A company puts a product on the market, and the product proves to be very successful. Customers use the product and it's almost perfect, but it needs some changes to make it really fit the context in which it's used. The company considers this and provides a customer-specific version. At the same time, on the basis of customer feedback, the company realizes that several customer segments would be better served with a product focused for each segment. So, the company ends up with a multitude of significantly similar product versions.

At this point, the company realizes that many of the required changes must be implemented for most or even all product versions and that implementing the same change multiple times is really inefficient, time-consuming, and error prone. This often results in the creation of a platform from which the different products and customer-specific versions can be derived. This significantly improves development efficiency. However, a new challenge enters the arena: managing the points in the platform where the product versions' functionalities differ—that is, *variability management*.

Variability management involves two key challenges. First, industrial reality shows that for successful platforms, the number of variation points, variants (alternatives that can be selected for a variation point), and dependencies between variation points and variants easily reaches staggering levels. We've seen cases with tens of thousands of variation points. The sheer number of variation points often results in having to allocate a rapidly growing percentage of the R&D budget to resolve

44 IEEE SOFTWARE | PUBLISHED BY THE IEEE COMPUTER SOCIETY 0740-7459/15/$31.00 © 2015 IEEE

Feature Modeling of Two Large-Scale Industrial Software Systems: Experiences and Lessons Learned

Daniela Lettner[1] Klaus Eder[2] Paul Grünbacher[3] Herbert Prähofer[4]

[1]Christian Doppler Laboratory MEVSS, Johannes Kepler University Linz, Austria
[2]KEBA AG, Linz, Austria
[3]Institute for Software Systems Engineering, Johannes Kepler University Linz, Austria
[4]Institute for System Software, Johannes Kepler University Linz, Austria

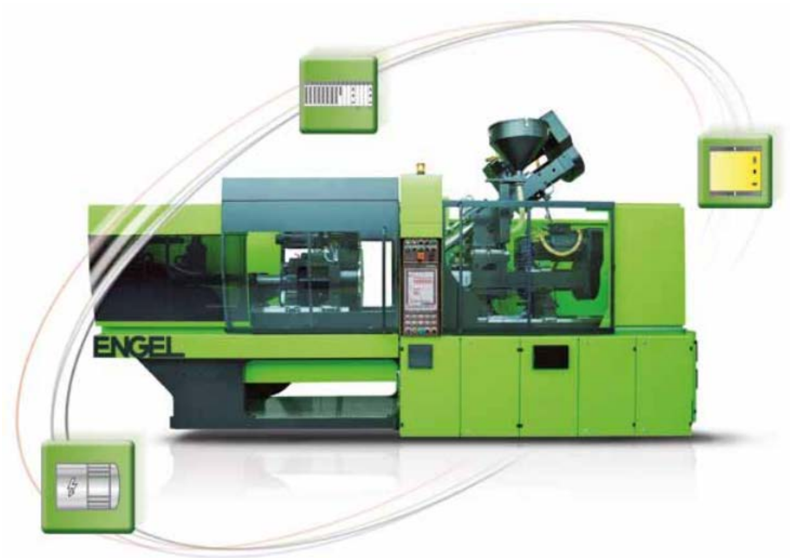
Invited Talk KV PLE – May 18, 2016 – JKU Linz

Large-scale industrial software product lines

KePlast

Automation solution for injection molding machines

3.8 million LoC



KeMotion

Control system solution for industrial robots

2.7 million LoC

Features models and features in industry:
modeling large-scale industrial systems remains hard

Distinguish **high-level** system features from **lower-level** capabilities

For what purpose do we need **feature models**?

What are the **dependencies** between feature models?

Berger, Lettner, Rubin, Grünbacher, Silva, Becker, Chechik, Czarnecki: **What is a feature?: a qualitative study of features in industrial software product lines.** SPLC 2015.

Related research

JOHANNES KEPLER UNIVERSITY LINZ

Megamodels / Macromodels

- Bézivin et al. MDAFA’04, OOPSLA/GPCE‘04: Global model management
- Salay et al. ASE‘08, MODELS‘15: Managing multiple models at high abstraction levels
- Simmonds MODELS’15: Megamodels supporting software process modeling

Modeling in-the-large, dependency and consistency management

Feature modeling in practice

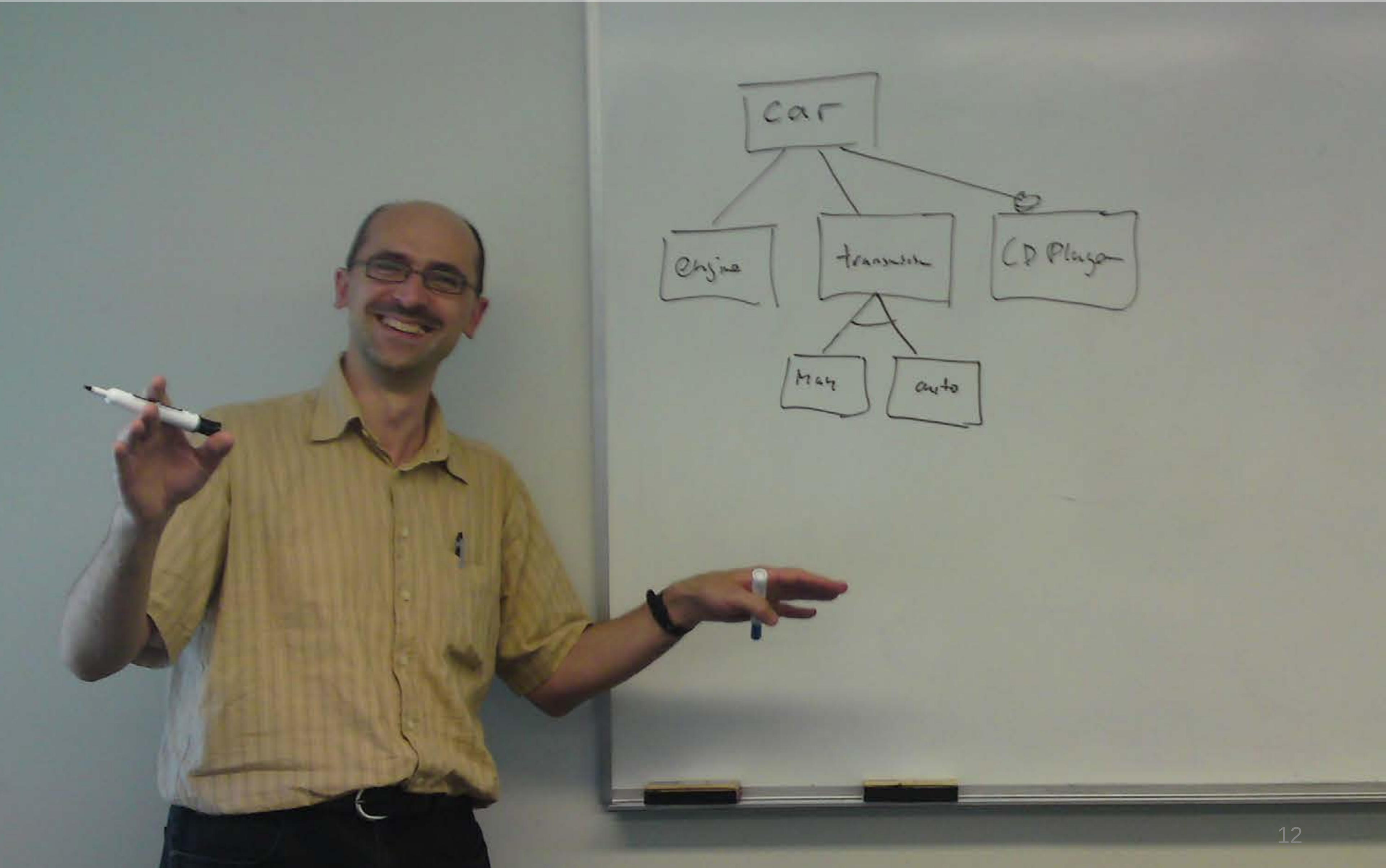
- Kang et al. ANN. SOFTW. ENG. 1998: Feature spaces
- Lee et al. SPLC‘00: Modeling experiences related to elevator control SPL
- Berger et al. SPLC‘15: Analysis of features in real-world settings

Only few reports exist on feature modeling in large-scale systems

Objectives guiding our work

- Developing **initial** feature models of KePlast and KeMotion
- Understanding **use** of features and feature models
- Investigating **purpose**, **scope**, **granularity**, and **dependencies**

Professor, what is a feature?



we found 35 definitions of “feature”

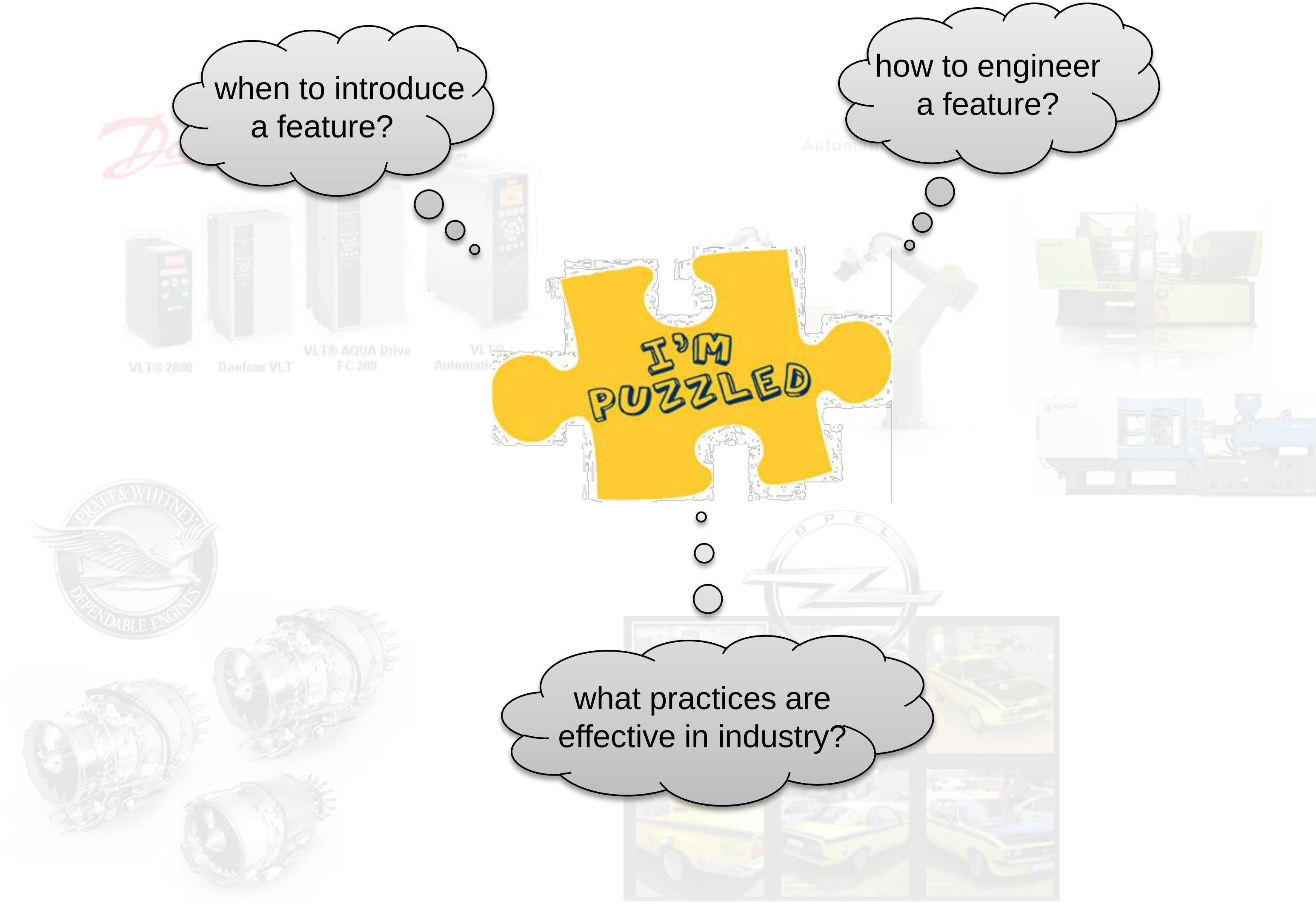
A feature represents an aspect valuable to the customer. [Riebisch03]

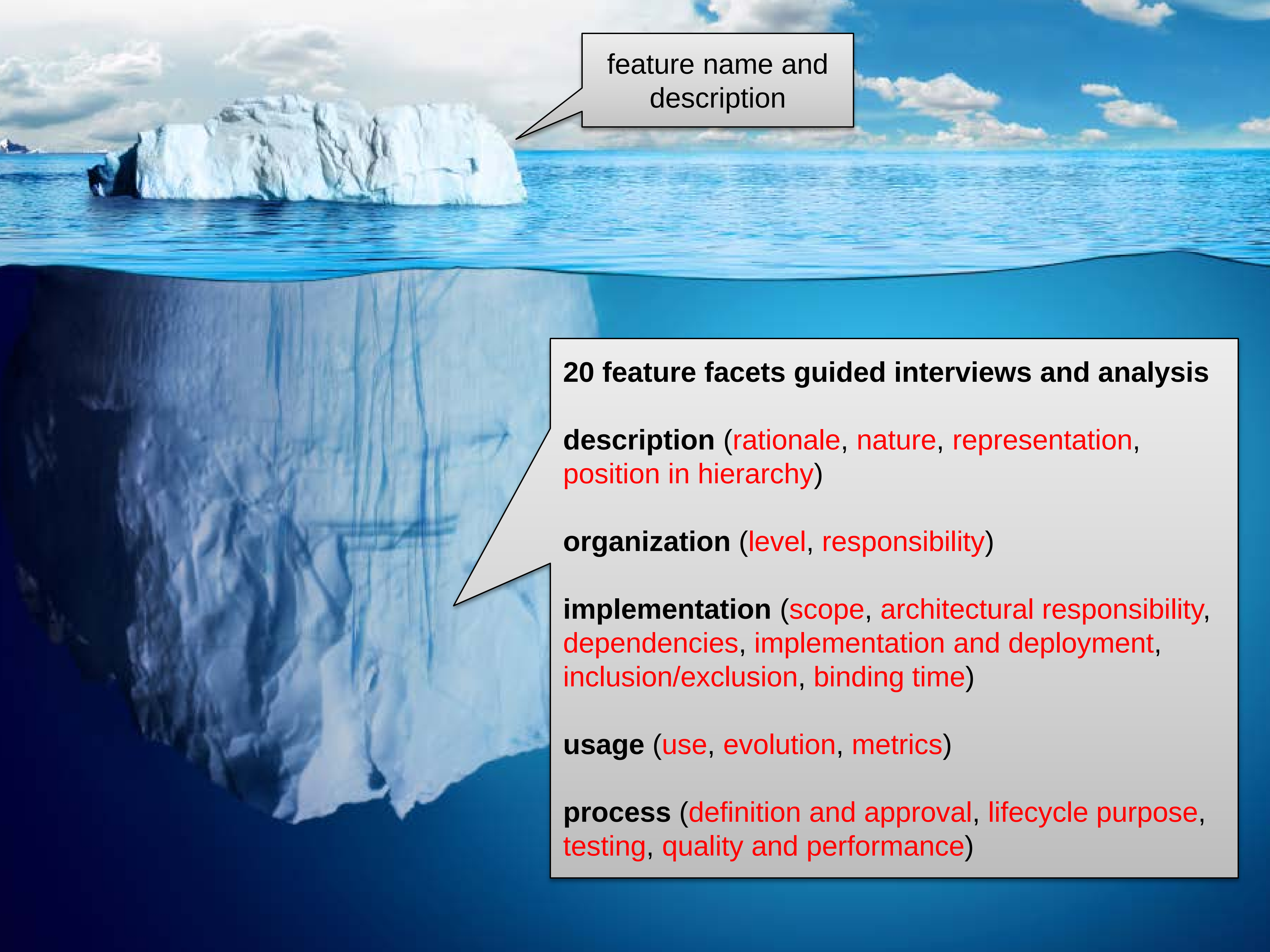
A Feature F of a product P is a product requirement $R \subseteq D$ that is visible to a user of the product P. [John10]

A feature is an increment of functionality, usually with a coherent purpose. [Zave99]

Customers and engineers usually speak of product characteristics in terms of the features the product has or delivers. [Kang++02]

features in industry





feature name and description

20 feature facets guided interviews and analysis

description (rationale, nature, representation, position in hierarchy)

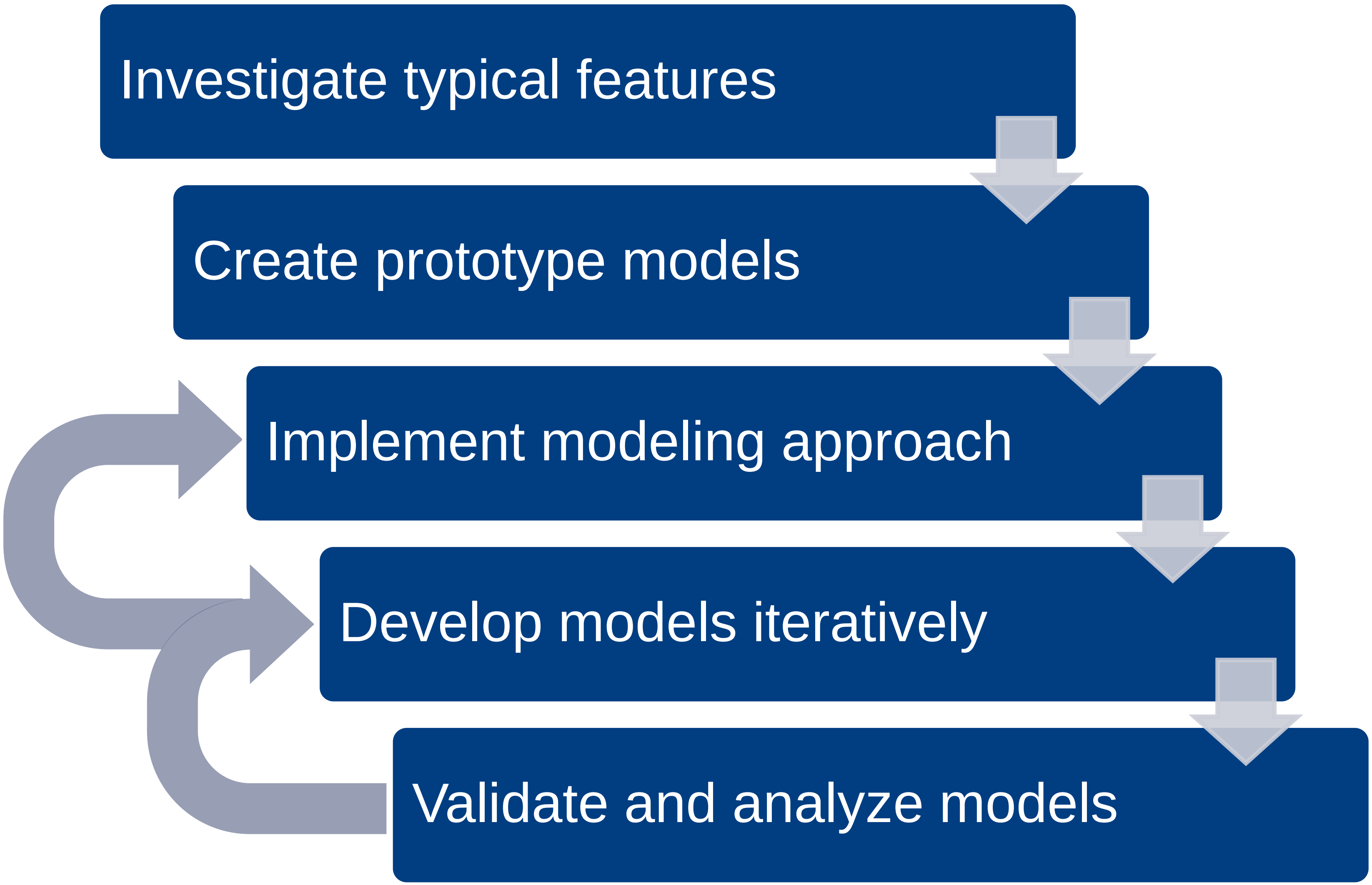
organization (level, responsibility)

implementation (scope, architectural responsibility, dependencies, implementation and deployment, inclusion/exclusion, binding time)

usage (use, evolution, metrics)

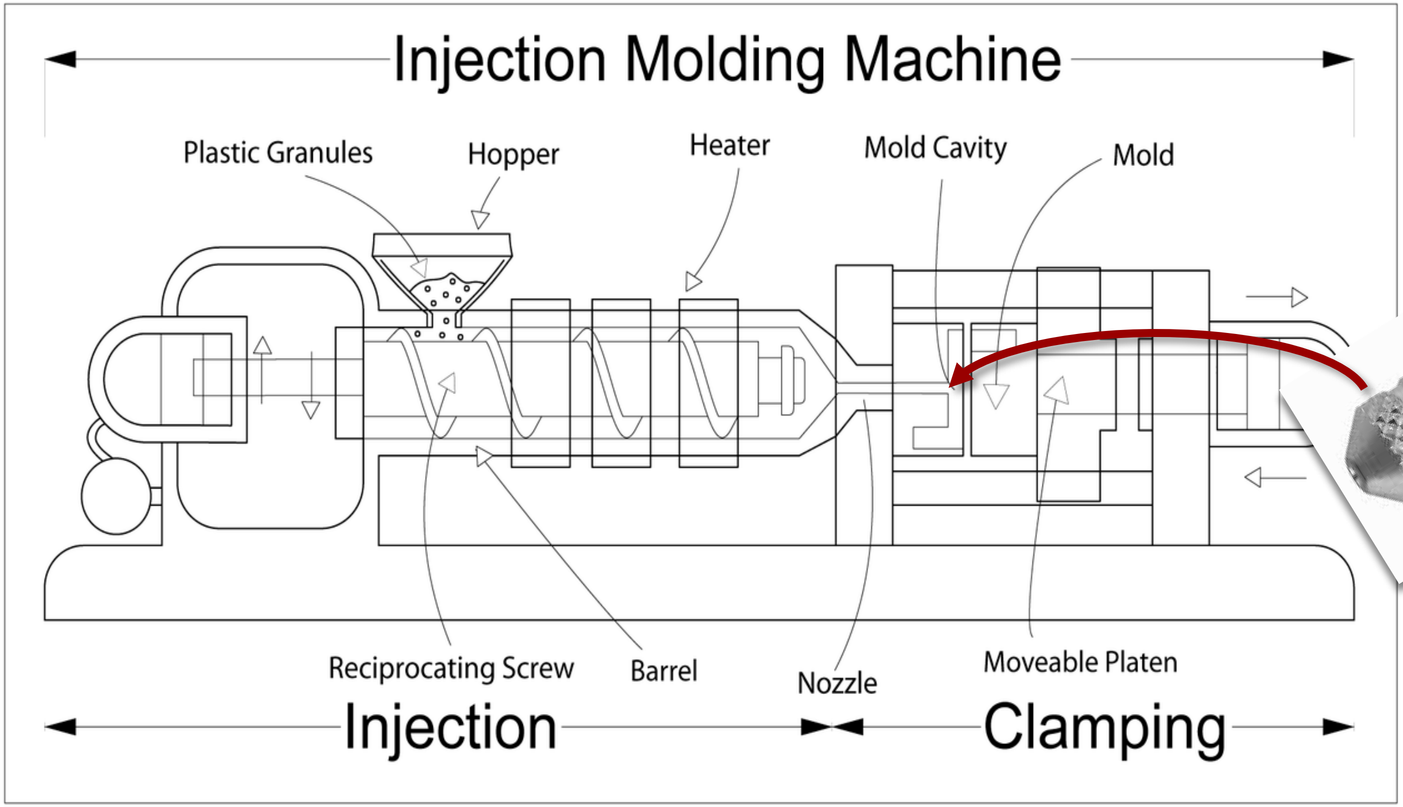
process (definition and approval, lifecycle purpose, testing, quality and performance)

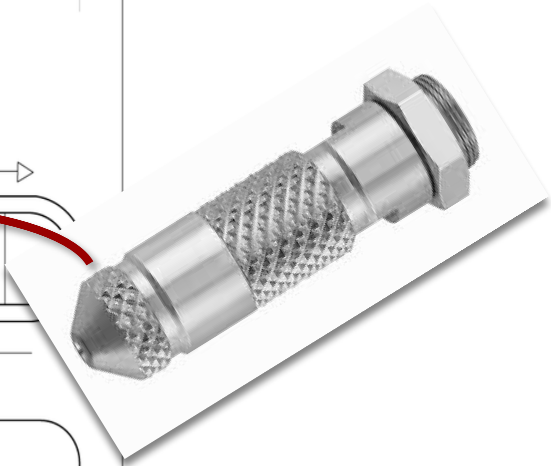
Preparatory steps and modeling process



The **MoldCavityPressureSensor** feature

Provides a **quality index** of an injection-molded part (i.e., reproducibility of pressure characteristic)





EXAMPLE

Problem space representation

JKU JOHANNES KEPLER UNIVERSITY LINZ

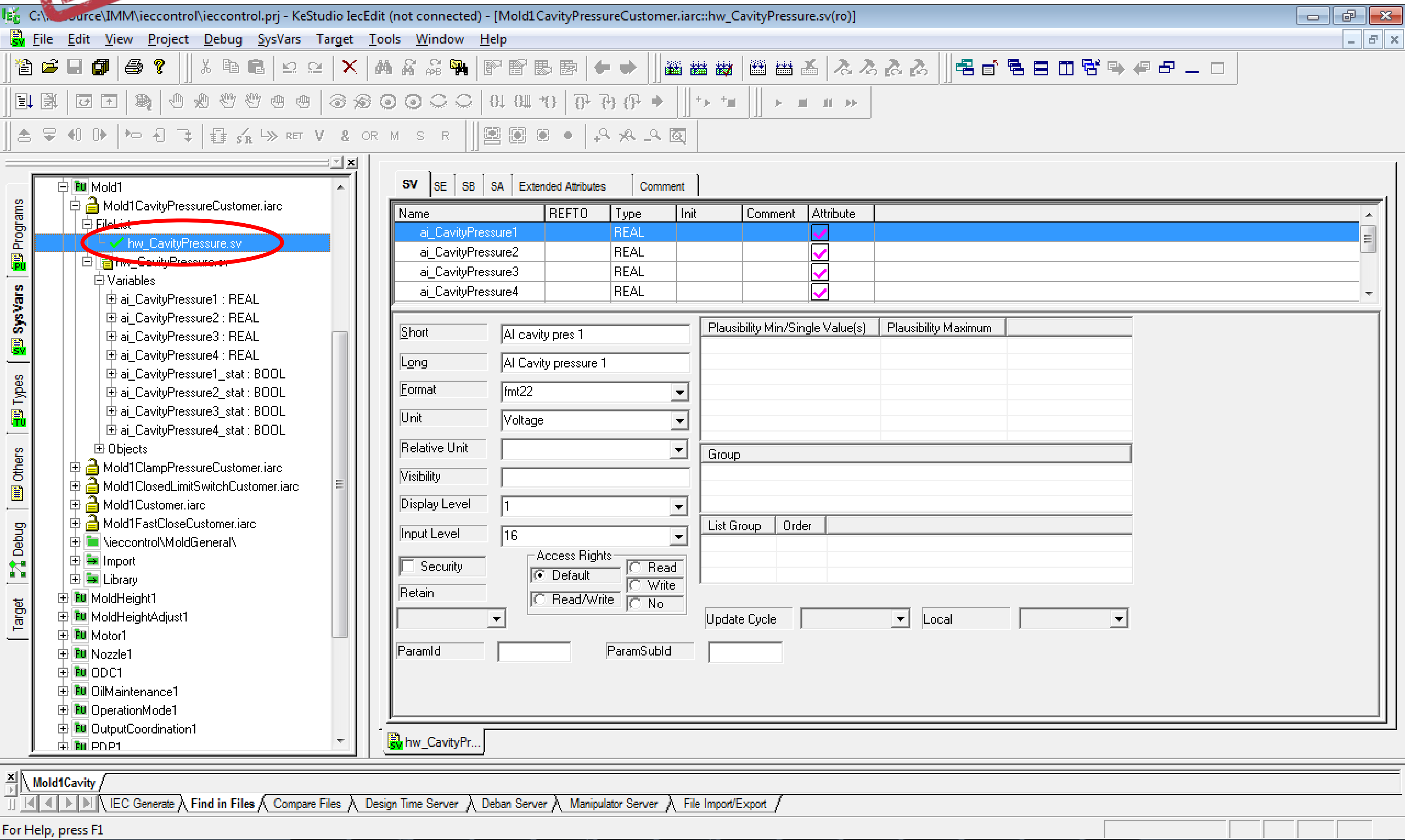
	Topic	Feature	Internal remarks	KePlast i1000 Eco	KePlast i1000	KePlast i2000	KePlast i5000	Reference to Ord. no.	planned in Version	Option
6										
7	Control hardware									
97	Clamp									
98		Movement ramp control by dynamic position feedback		x	x	x	x			
99		Mold protect by time control		x	x	x	x			
100		Mold protect by dynamic clamp force control	via force sensor			x	x			
101		Support of clamp force transducer			x	x	x			
102		Automatic clamp force control	protection against over-injection because clamp force is too low and mold can not be closed excl. sensor		po	po	po	on request	>1.3 2	x
103		Support of mold cavity pressure sensor	cut off trigger on mold cavity press.		o	o	o	83705		x
104		Loadable toggle lever curve	loading per file	x	x	x	x			
105		Rotary table (2, 3, 4 or 6 working stations) for V-IMM	Rotary or slide table	x	x	x	x		1.35	
106				-	-	-	-			
107	Mold heigth adjust									
108		Automatic mold height adjust for toggle machines		x	x	x	x			
109		Automatic mold height adjust for direct clamping machines		x	x	x	x			
110				-	-	-	-			
111	Ejector									
		Movement ramp control by dynamic position feedback		x	x	x	x			

documentation | Features | Options | Hardware Features

EXAMPLE

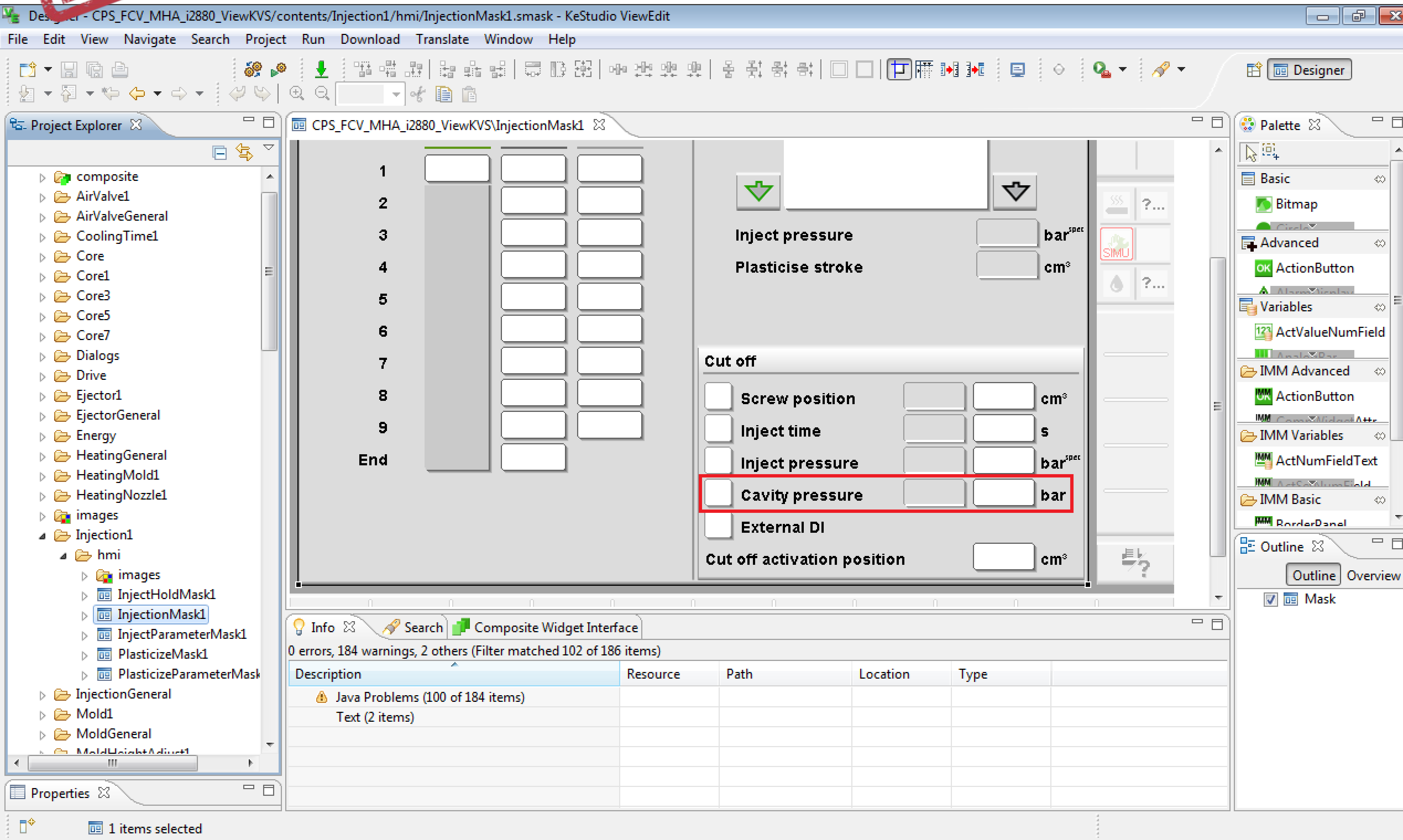
Solution space representation

JOHANNES KEPLER UNIVERSITY LINZ



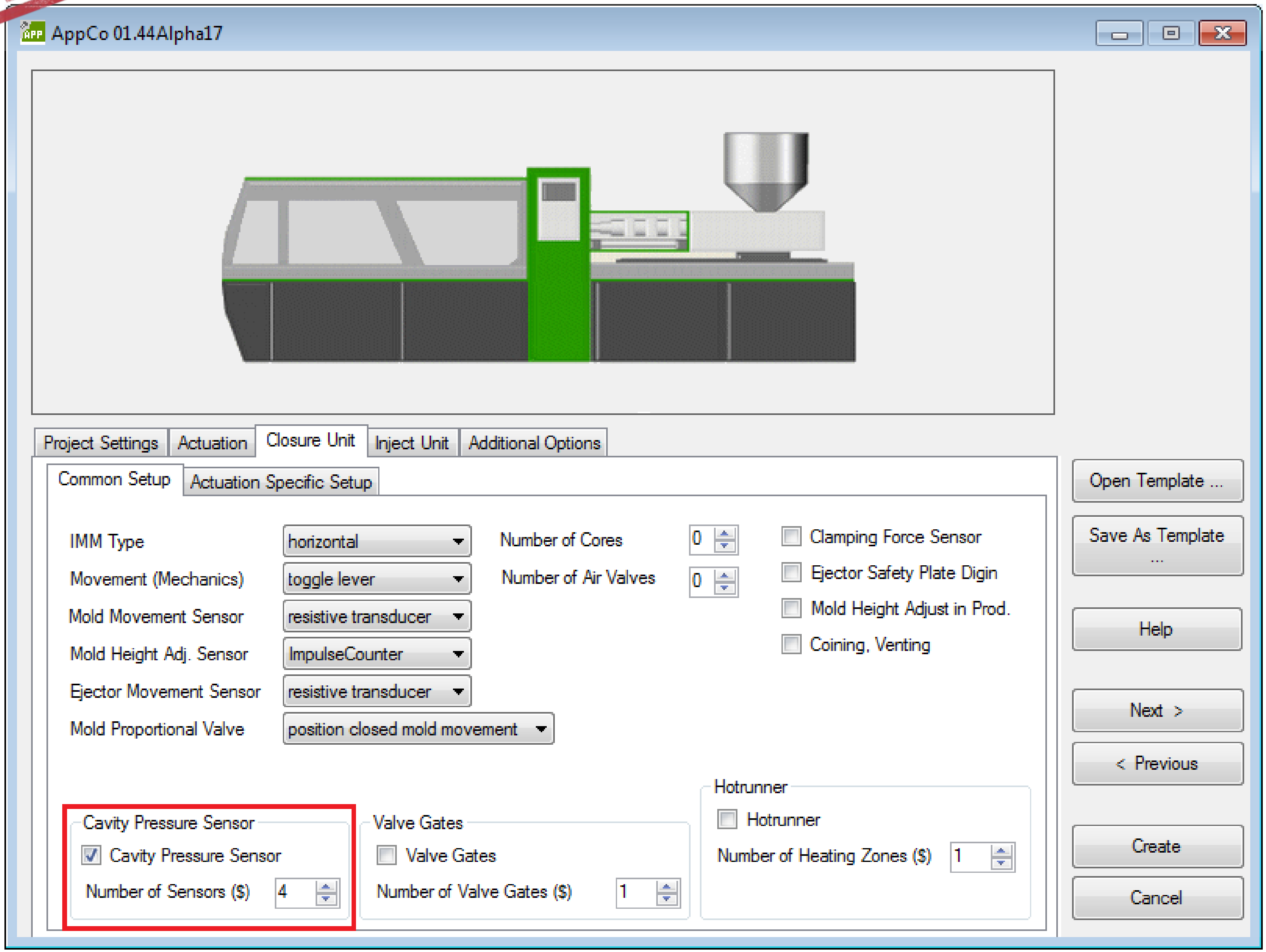
EXAMPLE

Solution space representation



EXAMPLE

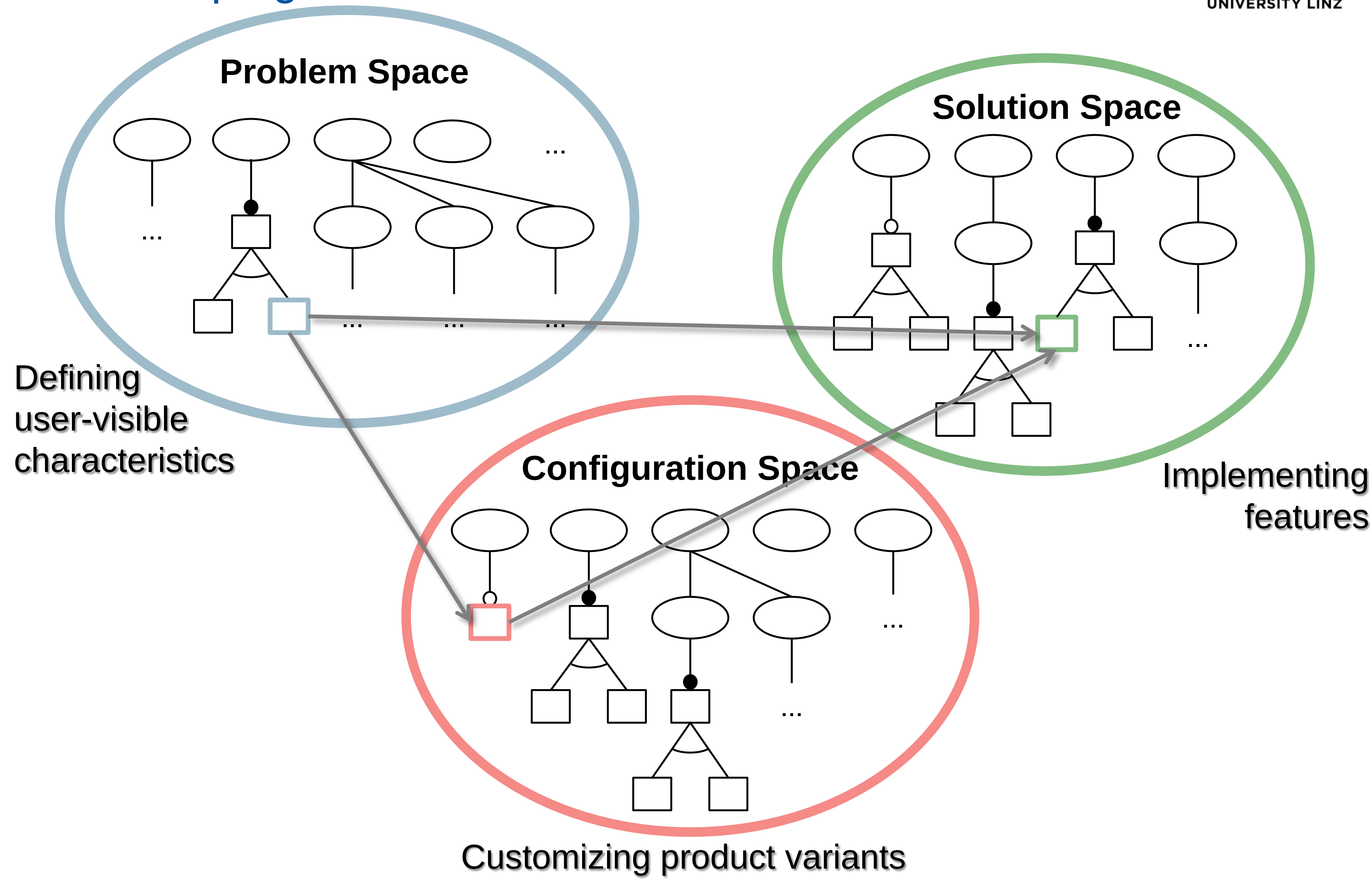
Configuration space representation



The **MoldCavityPressureSensor** feature

- Exists in different **forms** and for different **purposes**
- Hidden in **heterogeneous** tools and artifacts
- Occurs in **problem** space, **configuration** space, and **solution** space
- Relations between modeling spaces are **not documented** explicitly

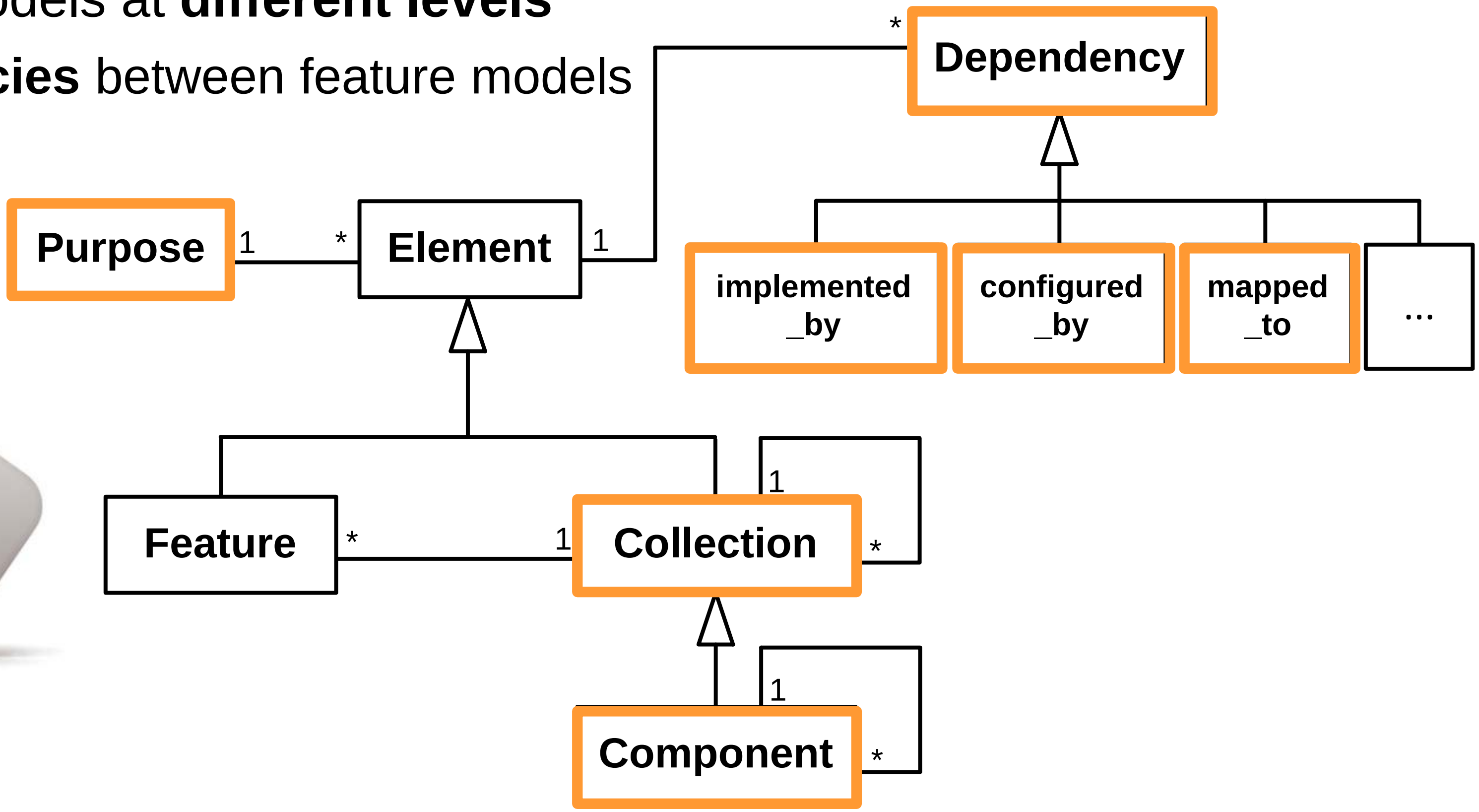
Developing interrelated hierarchical feature models



Modeling approach

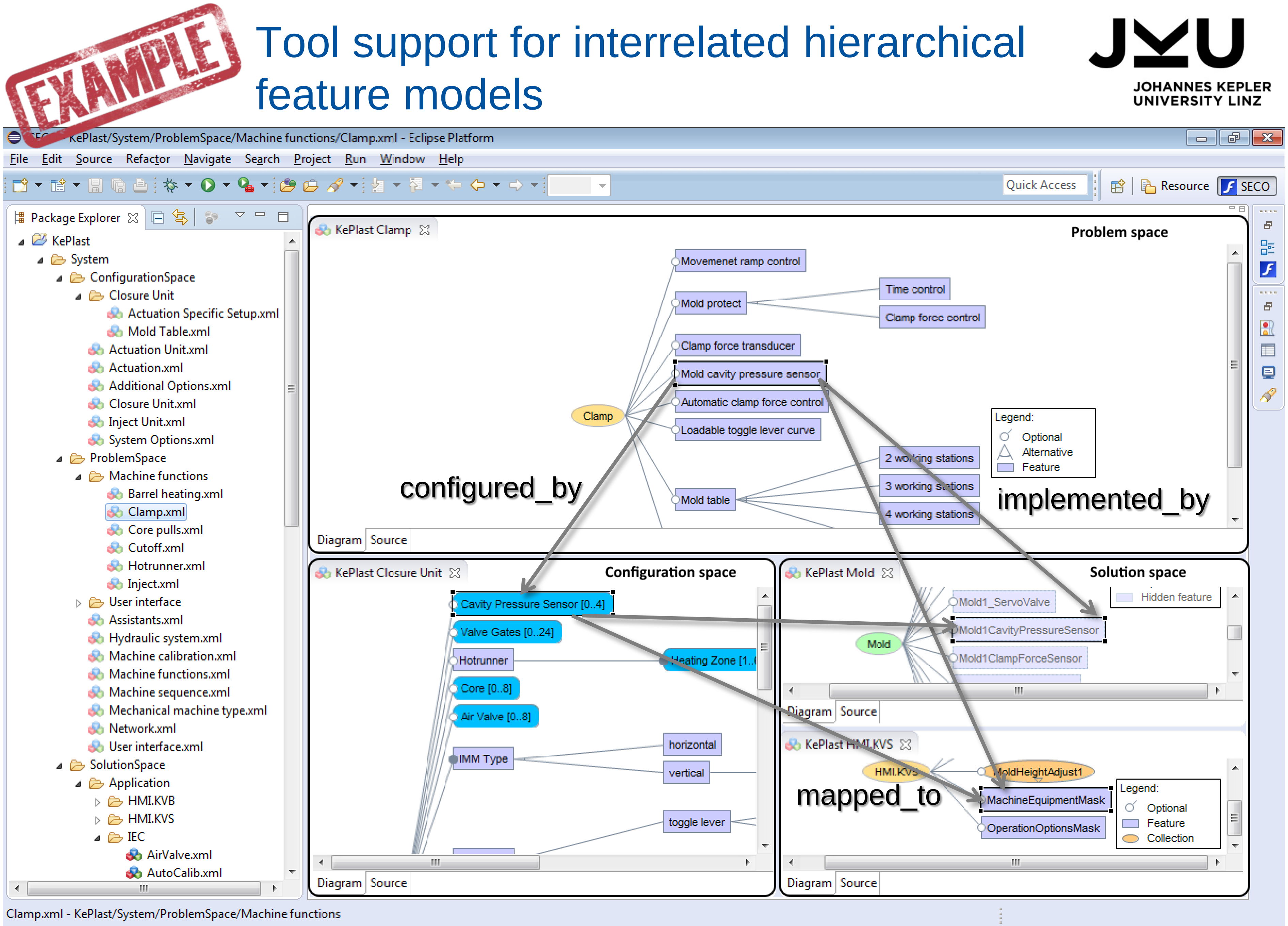
We extended the FeatureIDE tool suite:

- Feature models for **multiple purposes**
- Feature models at **different levels**
- **Dependencies** between feature models



Thüm, Kästner, Benduhn, Meinicke, Saake, and Leich. **FeatureIDE: An Extensible Framework for Feature-Oriented Software Development**. *Science of Computer Programming*, 79(0):70-85, January 2014.

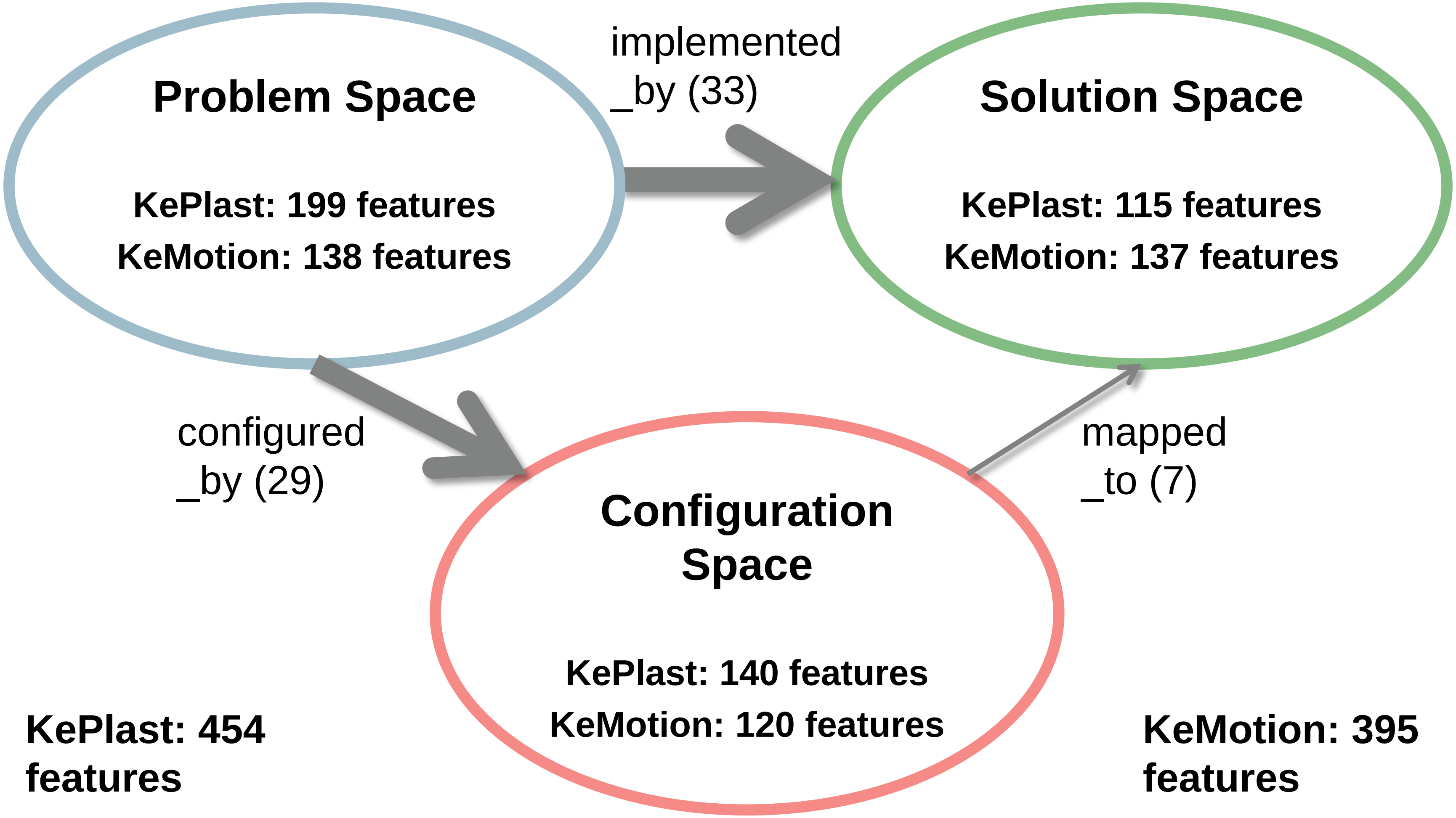
Tool support for interrelated hierarchical feature models



Top-down modeling strategy for both systems

	KePlast	KeMotion
Problem space	Product map spreadsheet	Commands for programming robots
Configuration space	Configuration tool AppCo	KeStudio configurator and MotionWizard
Solution space	KePlast core (IEC 61131-3) Visualization system	TeachTalk DSL Software architecture
Dependencies	Commonly known constraints	

System-wide model characteristics



KePlast: 454 features

KeMotion: 395 features

#1. Be specific about the **purpose** and **level** of features to facilitate the modeling process.

#2. Focus on **dependencies** between feature models to develop a **system-wide perspective**.

#3. Provide **code-level views** on the features.

#4. Use feature models to **limit variability**.

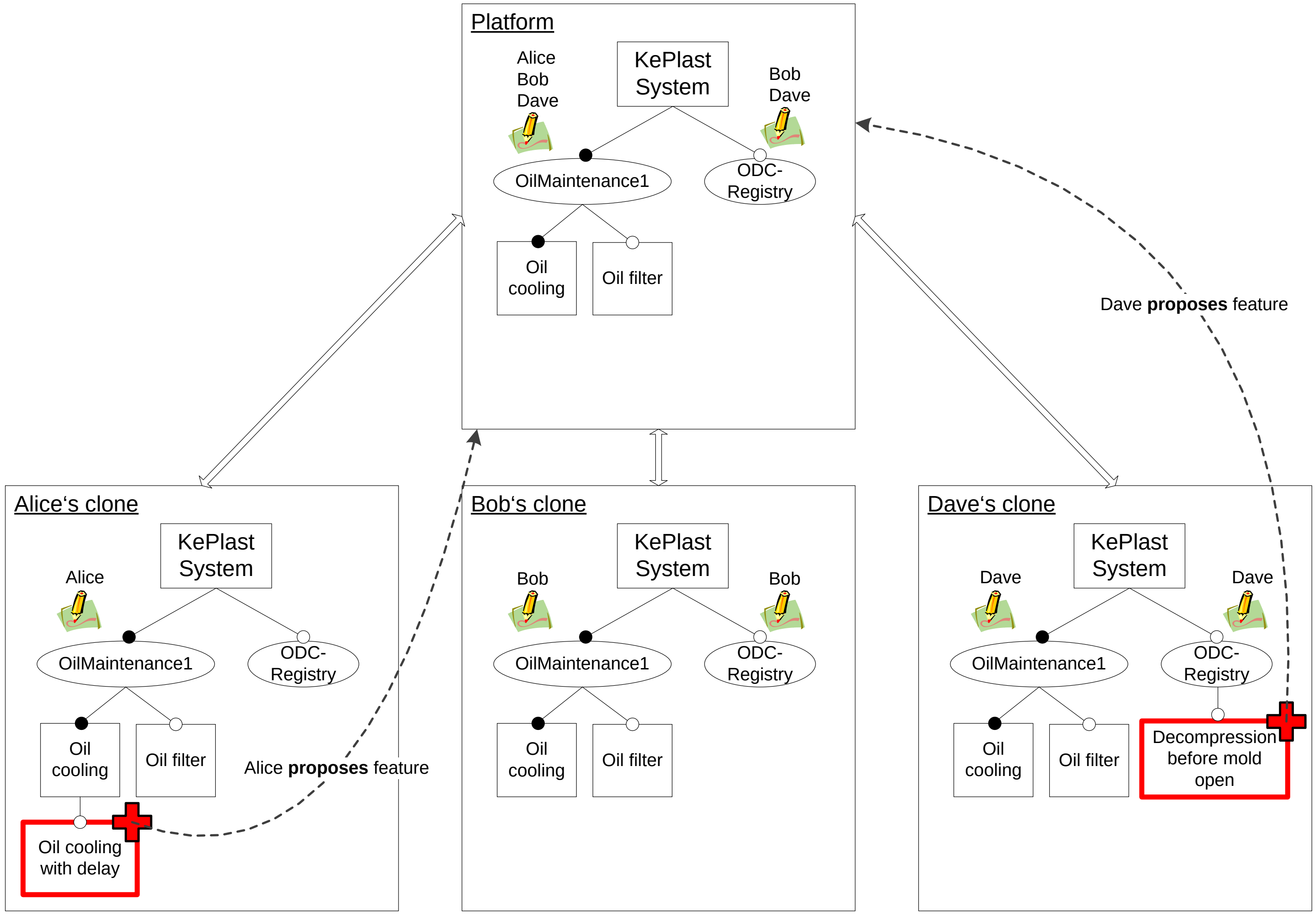
Improving Awareness of Emerging Features via Collective Developer Feedback

Daniela Rabiser[1] Jürgen Musil[2] Angelika Musil[2] Stefan Biffl[2] Paul Grünbacher[1]

[1]CDL MEVSS, ISSE, JKU Linz, Austria
[2]CDL-Flex, TU Vienna, Austria

Invited Talk KV PLE – May 18, 2016 – JKU Linz

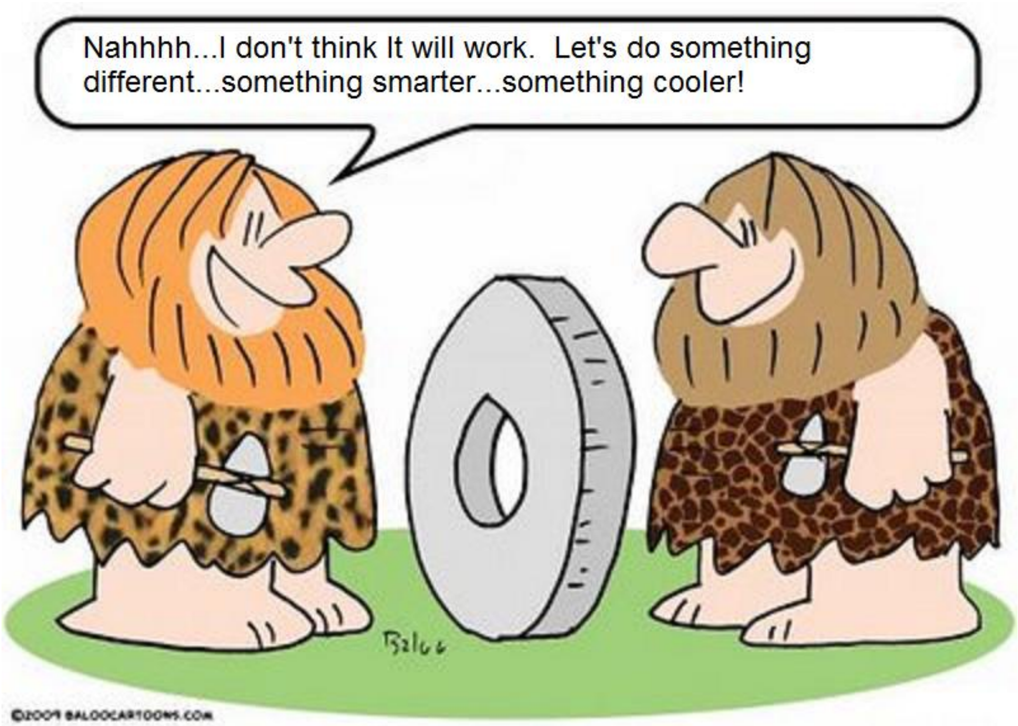
Motivation and Goal



<u>**Our Goal:**</u> collectively *aggregate* and *share* customer-specific features with distributed engineers

Re-inventing the wheel

- **Commonalities and variability** of products created via clone-and-own reuse approach are **poorly documented**



Lettner, Angerer, Prähofer, and Grünbacher. *A case study on software ecosystem characteristics in industrial automation software*. ICSSP 2014.

Re-inventing the wheel

- **Commonalities and variability** of products created via clone-and-own reuse approach are **poorly documented**

AE **B**: *“Typically only the AEs who created the products are aware of potentially interesting features.”*

AE **E**: *“To understand what has been developed, I have to look through a pile of changes.”*

Lettner, Angerer, Prähofer, and Grünbacher. *A case study on software ecosystem characteristics in industrial automation software*. ICSSP 2014.

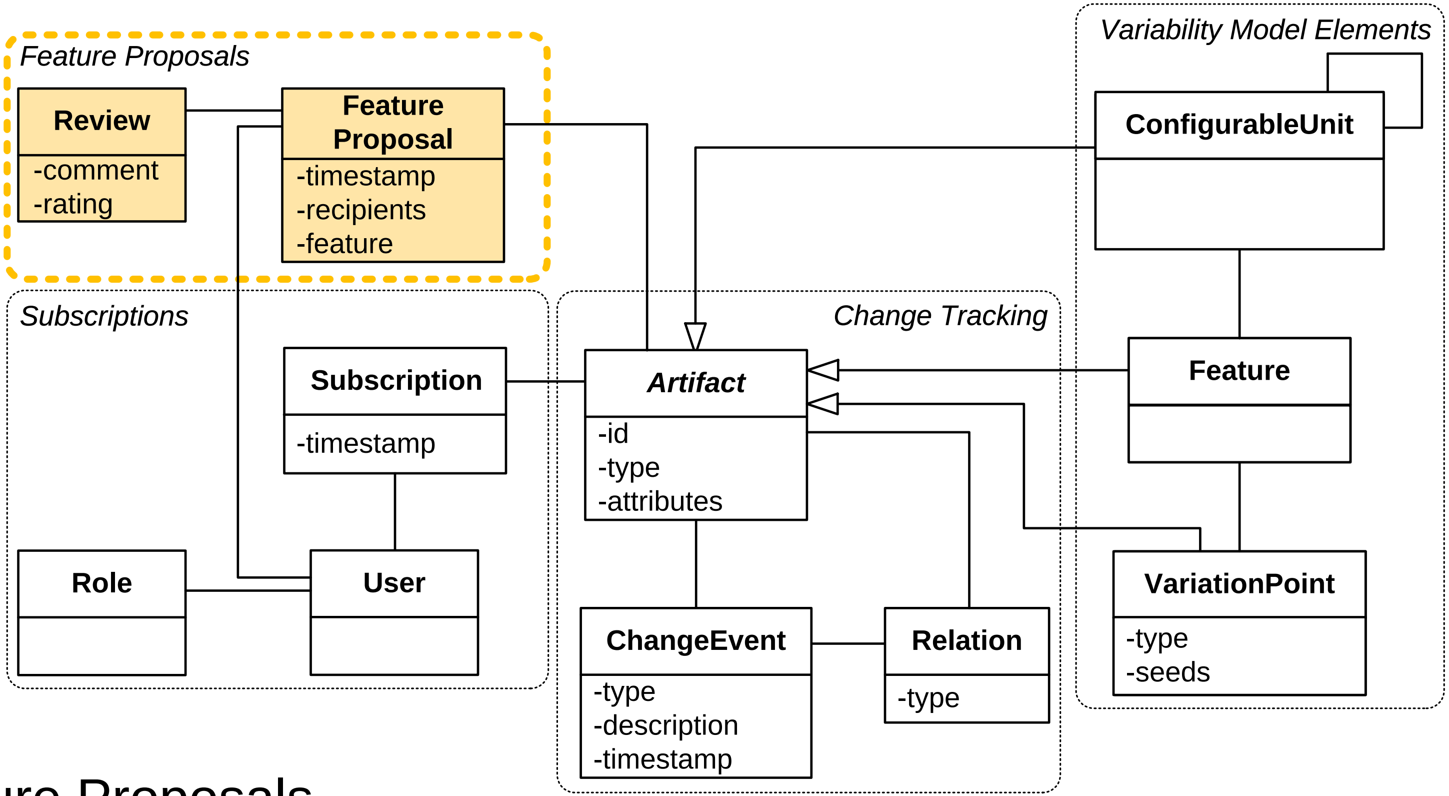
Feature Feed Approach

- **Improving developer awareness**
  - Documenting and propagating knowledge of developers regarding implemented features
  - Interested developers subscribe to features and receive notifications about changes

- **Supporting reactive evolution**
  - Making project-specific developments with high reuse potential visible
  - Developers propose locally developed features for integration into the PL platform

→ Supported via distributed **SECO awareness models (SEAMs)**

SECO awareness model (SEAM)



Feature Proposals

- Make valuable project-specific developments visible to other contributors
- Developers propose locally developed features for integration into platform

SEAM Views

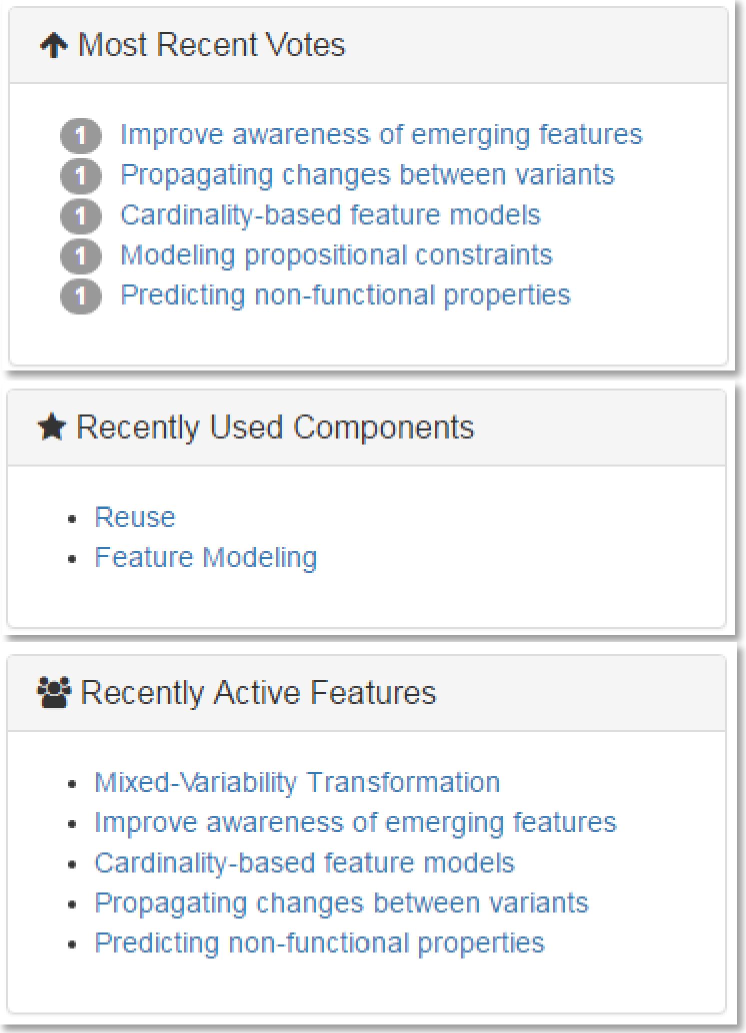
- Foster awareness regarding new developments
- Support reactive evolution of PL platforms

- **Feature feed views**
  - notify interested developers about relevant change events
  - based on a user‘s subscriptions and change events

- **Feature backlog views**
  - contain prioritized lists of proposed features
  - allows reviewers to inspect suitability of features and implementations
  - reviewers decide on the integration of features into platforms

Collaborative Feature Backlog

- Fosters **awareness** regarding new features
- Proposed features are **inspected** by **reviewers**
- **Features** are **prioritized** based on **votes**
  - Most Recent Votes
  - Recently Discussed Features
  - Recently Active Features
- **Notifications** and **Recommendations** per Email
- **Tags** for grouping features
- Extended Feature **Search**
- **Feature Metrics**



Inspecting trends and recommended features

Recent Activities

Daniela Rabiser voted on the feature Mixed-Variability Transformation
02 May 16:05

Daniela Rabiser edited the feature Improve awareness of emerging features
02 May 16:04

Florian Angerer proposed a new feature Mixed-Variability Transformation
02 May 15:59

Daniela Rabiser edited the feature Cardinality-based feature models
02 May 15:57

Daniela Rabiser edited the feature Improve awareness of emerging features
02 May 15:56

Daniela Rabiser edited the feature Improve awareness of emerging features
02 May 15:56

Daniela Rabiser edited the feature Propagating changes between variants
02 May 15:56

Daniela Rabiser edited the feature Predicting non-functional properties
02 May 15:55

Most Recent Votes

- 1 Improve awareness of emerging features
- 1 Propagating changes between variants
- 1 Cardinality-based feature models
- 1 Modeling propositional constraints
- 1 Predicting non-functional properties

Recently Discussed Features

- 1 Improve awareness of emerging features

Recently Used Components

- Reuse
- Feature Modeling

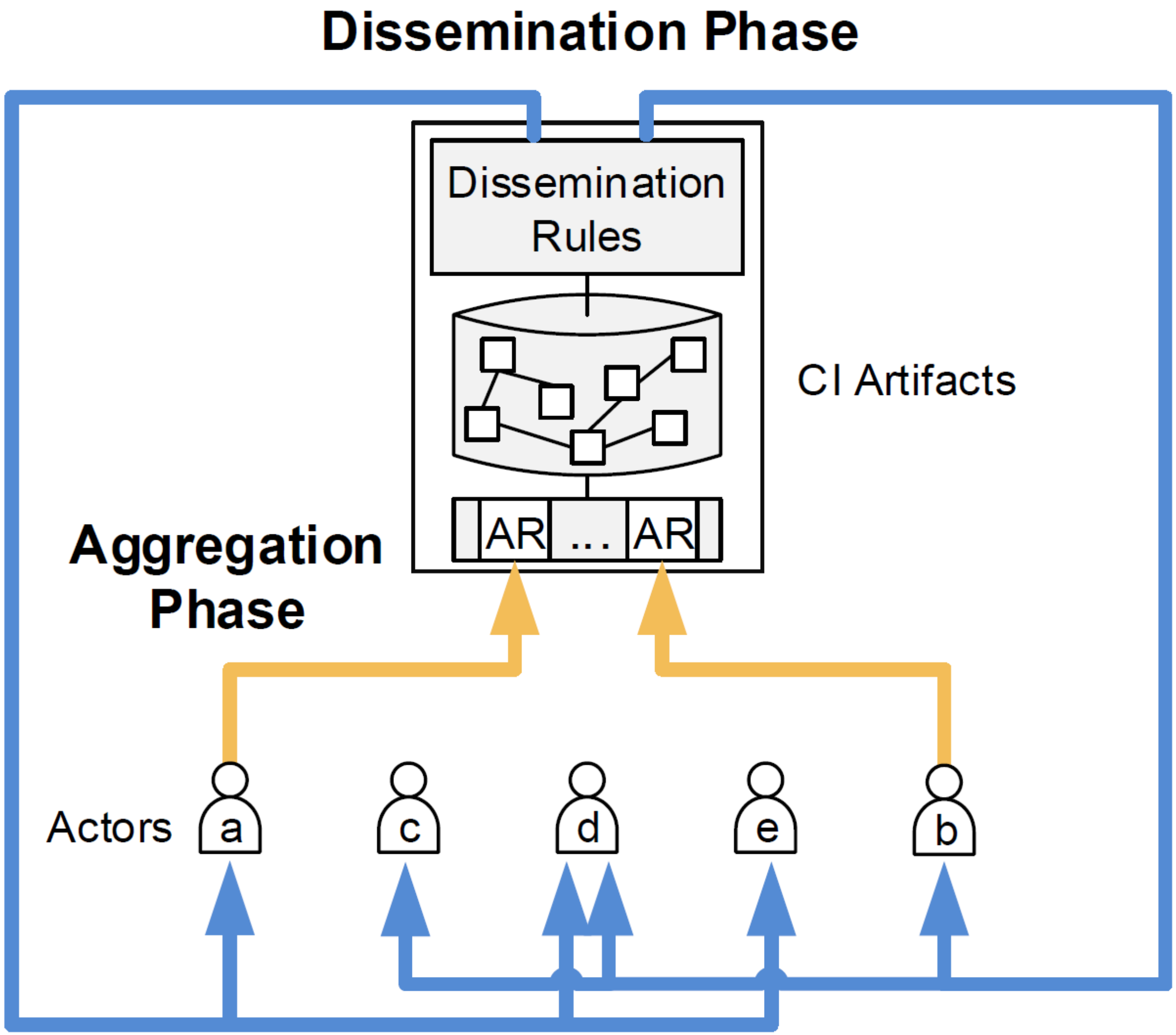
Recently Active Features

- Mixed-Variability Transformation
- Improve awareness of emerging features
- Cardinality-based feature models
- Propagating changes between variants
- Predicting non-functional properties

User Activities and Key Tool Capabilities

- **Proposing features**
  - Adding features
  - Specifying configurable units, related features and variation points
  - Updating feature proposals
  - Specifying feature tags
  - Changing feature states
- **Voting on features**
- **Discussing features**
  - Commenting on feature proposals
- **Inspecting trends and recommended features**
  - Investigating recommended features
  - Browsing through trending features and recent activities
- **Retrieving overview of features**
  - Inspecting feature metrics
  - Searching for features

Collective Intelligence Systems Process



User Study and Survey
to learn more about the usefulness of our tool

- @ **FOSD meeting 2016, 23-27 May:** Informal meeting to bring together researchers working on feature-oriented software development techniques – http://fosd2016.itu.dk/

- *Participant's task:* Propose new **features** that need to be supported by **future FOSD approaches** – https://fosd-fb.herokuapp.com/

- *Benefits and Value*
  - Collaborative documentation of future work that needs to be tackled by the FOSD community
  - Discussion of trending feature proposals at the end of the workshop
  - Provide and get detailed comments on proposed features (e.g., future work you plan to address)
  - Establish new collaborations with participants who identified similar features

Survey

- Learn more about usefulness of CFBP tool
  - Feature Metrics, Recommendations
- Use feedback to refine feature metrics and recommendations

- Investigate **usability** and **utility** of CFBP tool

User study and survey will be performed next week!