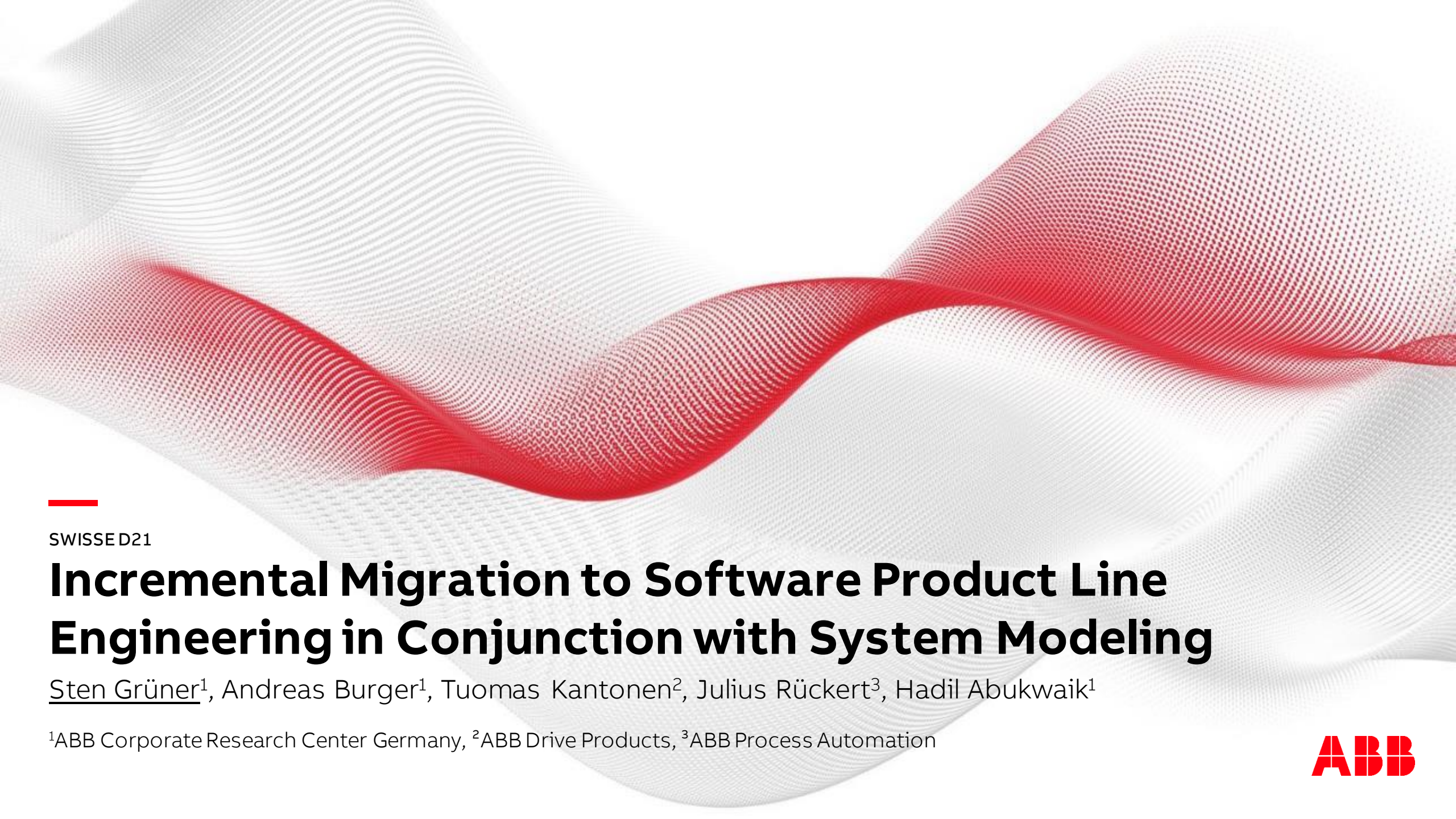


SWISSE D21

Incremental Migration to Software Product Line Engineering in Conjunction with System Modeling

Sten Grüner[1], Andreas Burger[1], Tuomas Kantonen[2], Julius Rückert[3], Hadil Abukwaik[1]

[1]ABB Corporate Research Center Germany, [2]ABB Drive Products, [3]ABB Process Automation

Well positioned across global markets

ABB is a leading global technology company that energizes the transformation of society and industry to achieve a more productive, sustainable future.

By connecting software to its **electrification**, **motion**, **process automation and robotics & discrete automation** portfolio, ABB pushes the boundaries of technology to drive performance to new levels

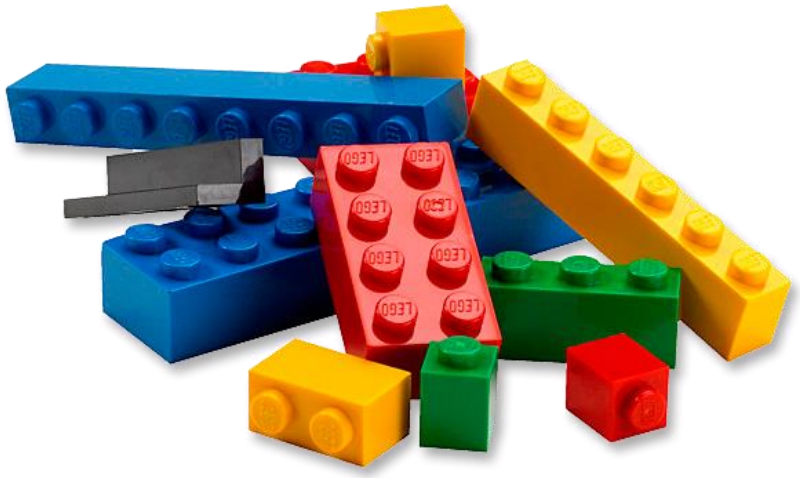
Variable Frequency Converters

- Hard real-time embedded system to control electrical motors
- Used in wide range of applications and industrial domains
- Additional variability introduced by power ranges and environment, e.g., network connectivity
- Low voltage portfolio (up to 690V) is just a fraction of ABB’s drives portfolio

Growing Variability Challenges

- Digitalization of industry (aka Industry 4.0) and Industrial IoT:
  - Higher complexity of the firmware (new features)
  - More customer-tailored products
- Firmware customization for an individual application, OEM and even single customer needs
- Systematic, strategic re-use of software components across device lifecycle needed

Software Product Line Engineering



“A product line is a set of software intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.”

But how to migrate existing agile product development to Software Product Line Engineering methodology?

Status Quo Assessment
Step 1

Firmware application and developments

- Running on multiple hardware platforms on top of a real-time operating system
- Few millions LoC of embedded C/C++ code
- Custom build tool chain
- Multiple software layers ranging from HW-abstraction to application layer on top
- Compile- and run-time variability binding mechanisms
- Agile development running based on Scaled Agile Framework (SAFe)

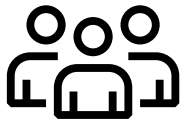
Virtual Platform Level Assessment

Level	Characteristics
L0	ad-hoc clone (pure clone-and-own)
L1	Structured repository to maintain clones, Keeping track of changes across variants, Clone management tool
L2	Identified features, and located in code, Modularized features for cloning, Mechanisms for cloning/including/ excluding features from clones
L3	Feature model per project, Feature configuration via build files
L4	Feature model across the project, Checking consistency of feature selections, Cloning selected features into variant
L5	Integrated SPL platform

Antkiewicz et al. – Flexible product line engineering with a virtual platform, SPLC 2014

Identify Stakeholders and Their Needs

Step 2



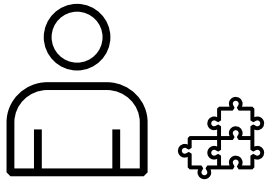
PRODUCT MANAGEMENT

- In charge of defining customer features within the feature model
- Assigning features to pro

Benefits:

- Direct work on the model
- Decision traceability

Feature model as primary interface

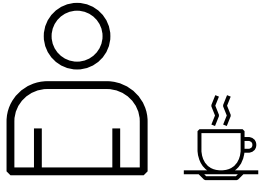


DOMAIN ARCHITECT

- Defi
  syste
- Defi

Benefits

- Alwa
- Documented build system flags



SOFTWARE DEVELOPER AND TESTER

- ble for the implementation
  ftware features
- nsumer of the feature model

- ent decision documentation
- Change impact traceability

Short-term benefits are reachable by a non-invasive SPLE usage along with existing tools and processes

Feature = customer-visible software feature creating value

Define a Non-Invasive SPLE migration Process

Step 3

DOCUMENT

- Non-invasive: truth is in the build toolchain
- Workflow: legacy
- Model: partial
- Costs: high fixed

SYNCHRONIZE

- Non-invasive: truth is in the build toolchain
- Workflow: legacy
- Model: partial, consistent
- Costs: low running

SIMULATE

- Non-invasive: truth is in the build toolchain
- Workflow: legacy and new
- Model: full, consistent
- Costs: high running

OPERATE

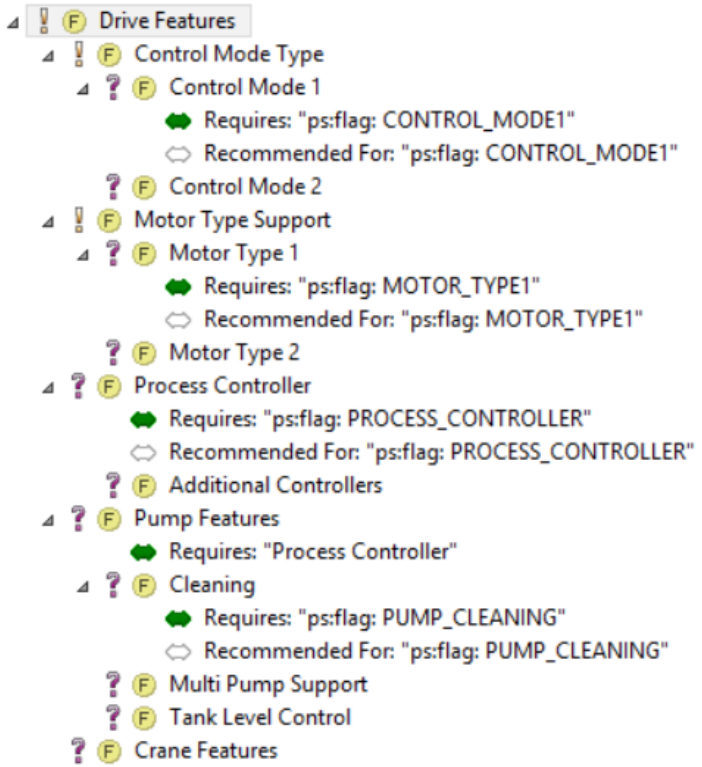
- Invasive: truth is in the SPL model
- Workflow: new
- Model: full, consistent
- Costs: organizational break needed

Time and effort required for adoption

Build a Proof of Concept
Step 4

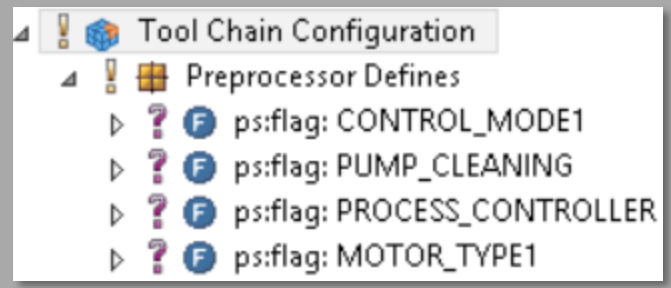
Problem space = Feature model

- 12 Variants
- Around 70 customer-facing features
- Maximum feature-tree depth of 4
- Mostly Boolean
- Constructed based on available public and internal documentation



Solution space = Family model

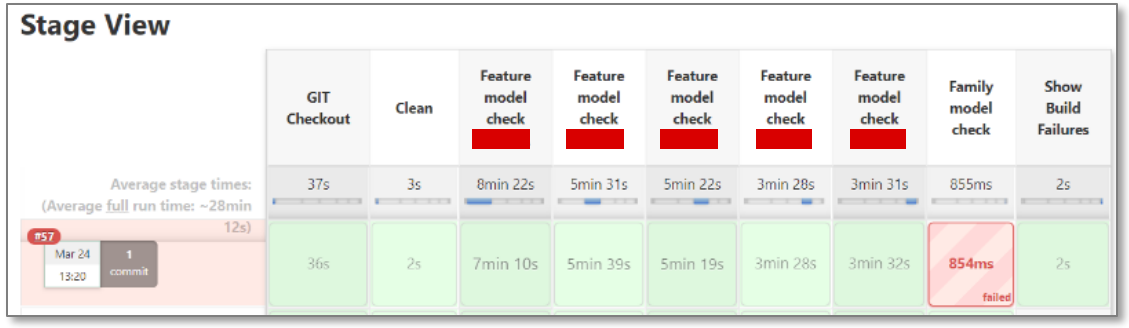
- Mostly based on pre-processor flags (more than 80)
- In addition to flags, menu-related properties were modeled



Feature Model Integration
Step 5

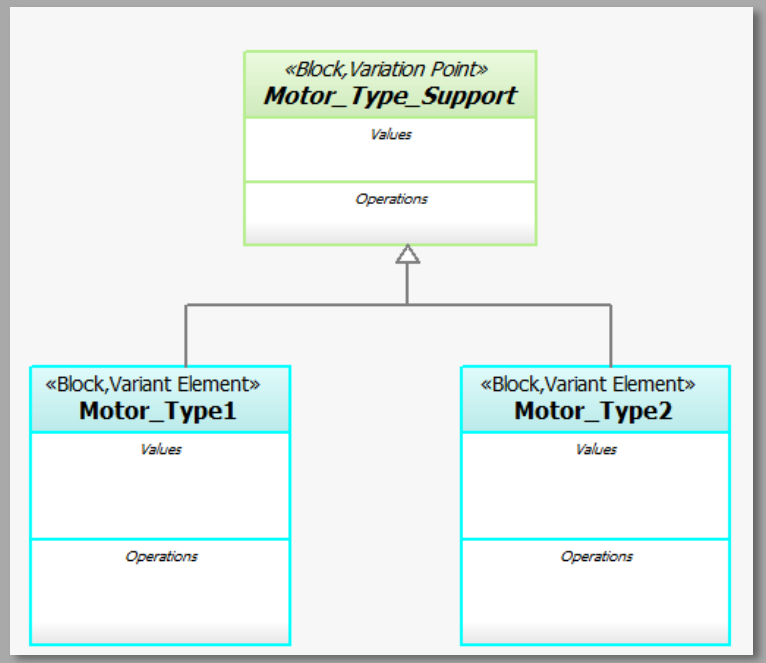
CI-integration for Synchronization

- Feature model is kept in git along with the source code
- Custom code to extract latest variant configuration
- Automatic comparison of build logs with the family model



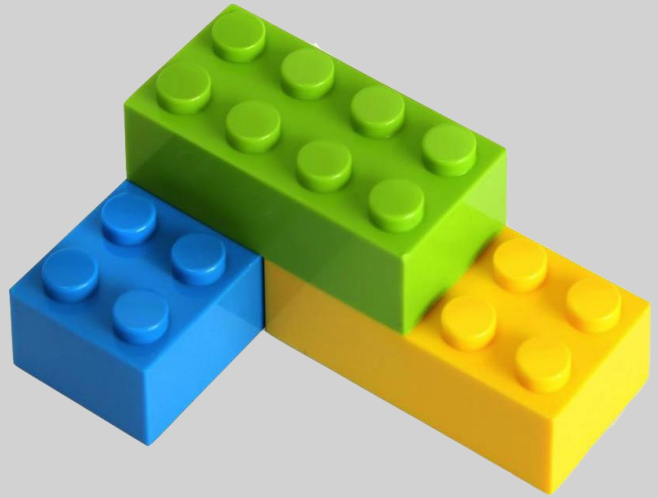
System Model Integration

- Based on SysML stereotypes
- Preprocessing functionality of pure::variants is used



Lessons Learned

- Non-invasive SPLE brings first benefits fast
- Involvement of product management was key to overcome organizational hurdles
- A roadmap of non-invasive introduction + commercially supported tools were essential for acceptance
- Commercial off-the-shelf SPLE tools have limited support for custom brownfield environments
- Solution space model design is highly use-case-specific: danger of over-engineering of the solution space
- Created SPLE demonstrator revealed that some existing binding mechanisms are not well-suitable for reuse



Dr.-Ing. Sten Grüner
Senior Scientist

Software Architecture Research Group
ABB Corporate Research Center Germany
Wallstadter Straße 59
68526 Ladenburg, Germany

sten.gruener@de.abb.com

Lessons Learned and Open Challenges

Lessons Learned

- Non-invasive SPLE brings first benefits while introducing
- Involvement of product management was key to overcome first organizational hurdles
- A roadmap of non-invasive introduction + commercially supported tools was essential for acceptance of SPLE methodology
- Commercial off-the-shelf SPLE tools have limited support for custom brownfield environments: watch out for available APIs!
- Solution space model design is highly use-case-specific: danger of over-engineering of the solution space
- Created SPLE demonstrator was abele to reveal that some existing binding mechanisms are not well-suitable for reuse

Open Challenges and Future Work

- Versioning of feature model itself and included features
- Not all aspects of solution space, e.g. tests, are covered yet
- Role-based access control to model was not yet evaluated
- Organizational costs and setup must be further evaluated especially for a “point of no return” between simulate and operate steps
- Quantified evaluation of benefits and Return on Investment

Placeholder for SysML extension discussion