Connecting Product Management and Software Architecture – Challenges for the Digital Transformation

PD Dr. Christoph Knieke / Prof. Dr. Andreas Rausch

Technische Universität Clausthal
Institute for Software and Systems Engineering
Clausthal-Zellerfeld, Germany
Email: christoph.knieke@tu-clausthal.de,
andreas.rausch@tu-clausthal.de

ComputationWorld 2021, April 18-22, Portugal

TU Clausthal

ISSE Institute for Software and Systems Engineering

Short resume of the presenter

PD Dr. Christoph Knieke

- 2004: Diploma in computer science
- 2011: Doctoral degree
- 2019: Habilitation degree

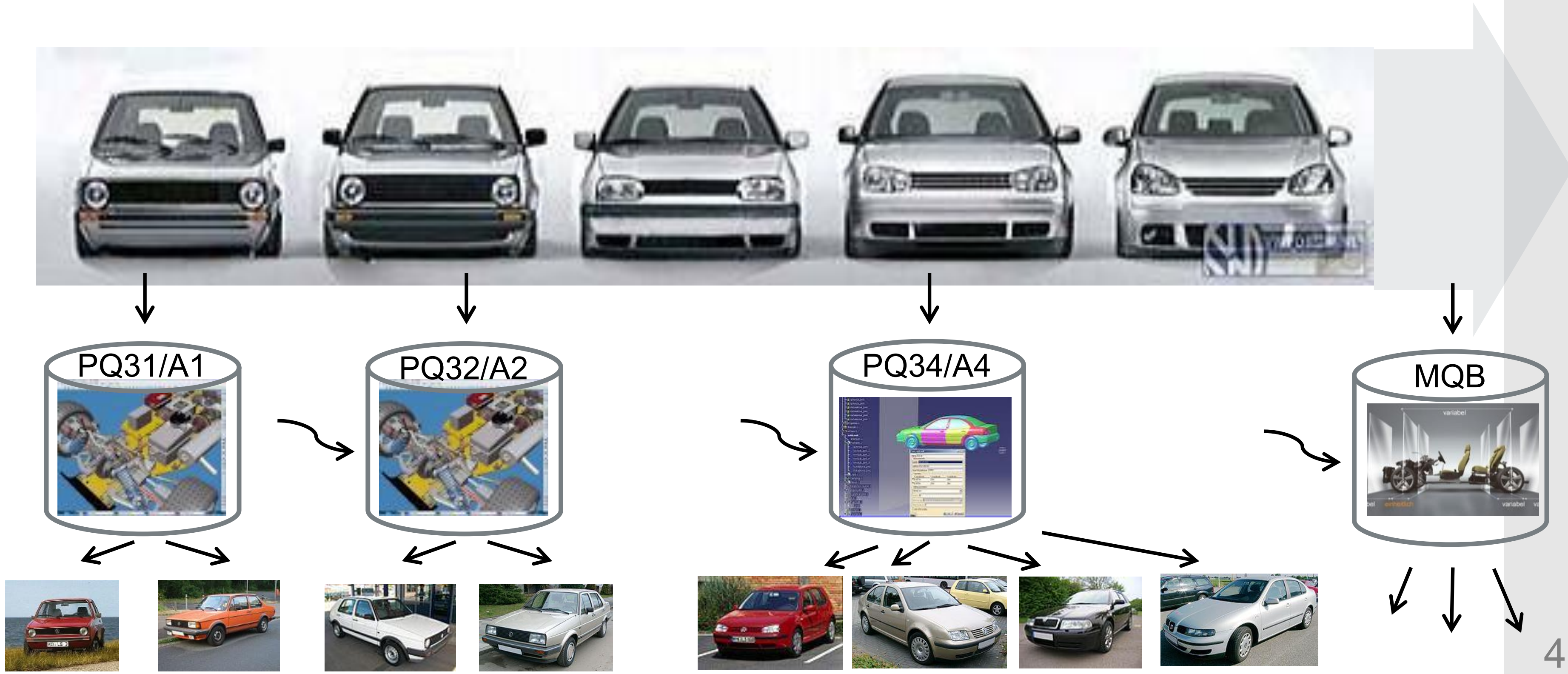
- Lecturer and post-doctoral researcher at TU Clausthal
- Leader of research group „Requirements, Architecture and Lifecycle Engineering at TU Clausthal
- Leader of several R&D projects, e.g., together with Volkswagen AG
- Organizer of special tracks at IARIA conferences

- Research interests:
  - Model-based systems engineering (MBSE)
  - Domain-specific modeling languages
  - Software product line engineering
  - Architecture evolution

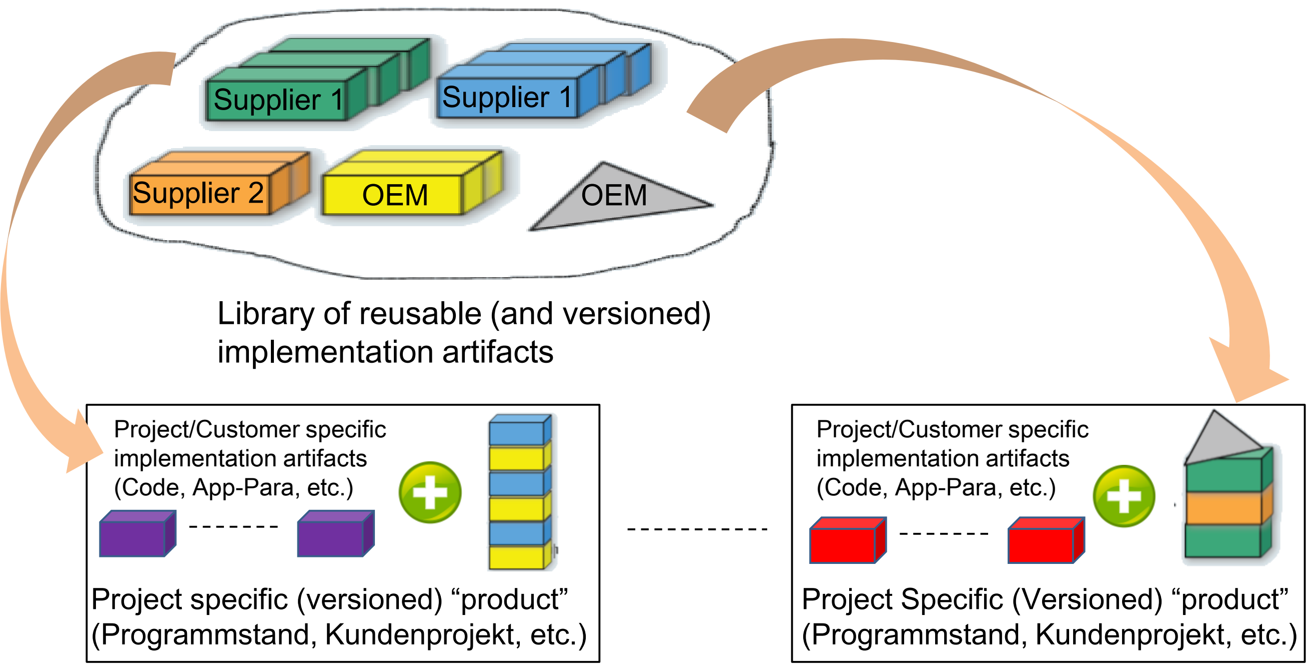
Contents

Platform Strategy – A Well-known Successful Approach in Automotive Industry

- Manufacturing of product families based on platforms since the 1960s
- High number of variants
- High degree of reuse



Software Sharing: A Platform Strategy for Software

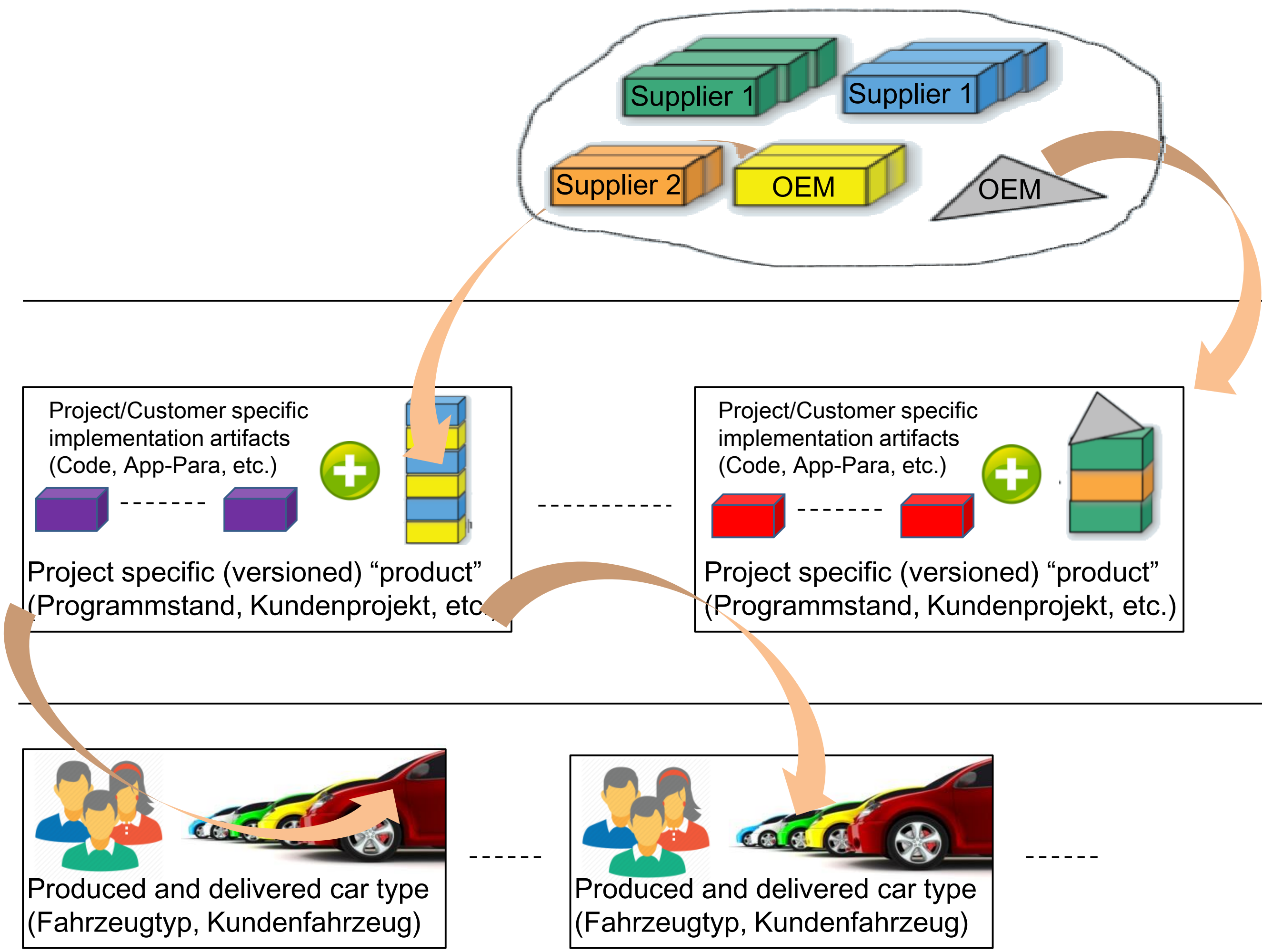


Benefits of Software Sharing

- Increasing of reusability
- Protecting the core know-how
- Saving of development costs
- Increasing of quality

Software Sharing = Essential Part of Platform Strategy + Success Factor!

Variability in Software Modules = Key Concept in Software Sharing

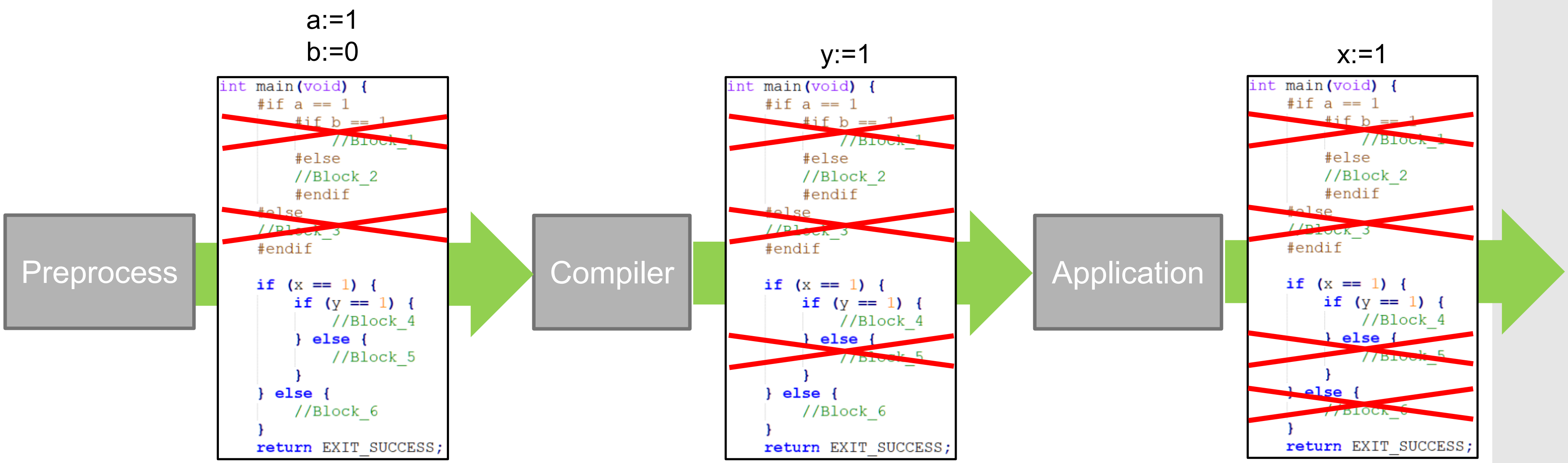


Configuration of **Positive Variability** (selection of implementation artefacts) and **Negative Variability** (system parameters conf. for preprocessing / compile)

Car type specific software application data set (**Negative Variability**), flashing, customer specific configuration (**Negative Variability** run-time!)

Configuring the Negative Variability

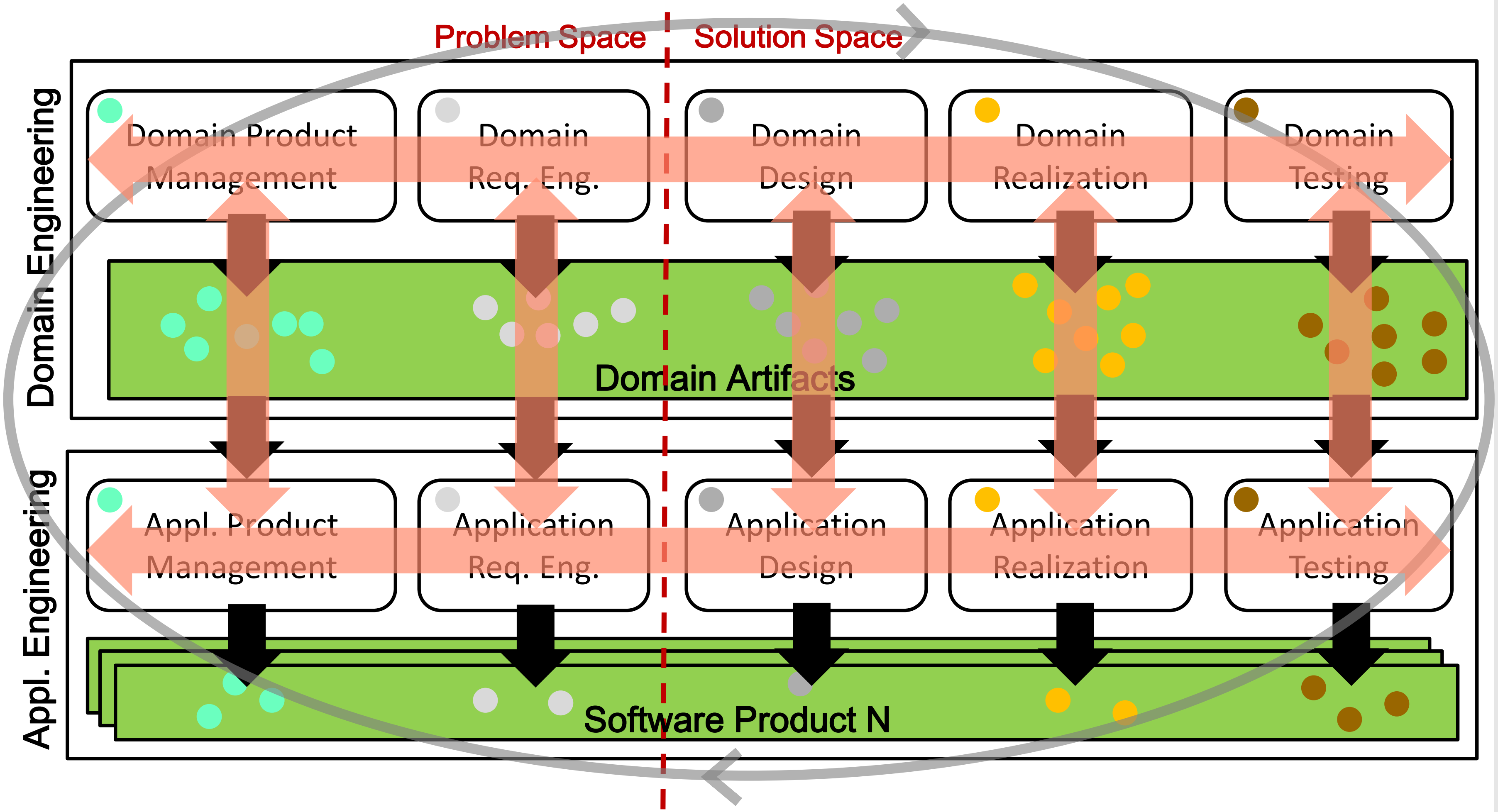
- By „switches“ in the code, preprocessing code is deleted
- Attention (partly compiler dependent):
  - Compilers can leave dead code in the program
  - Optimizing compilers can delete dead code
- Further parameter values set during application



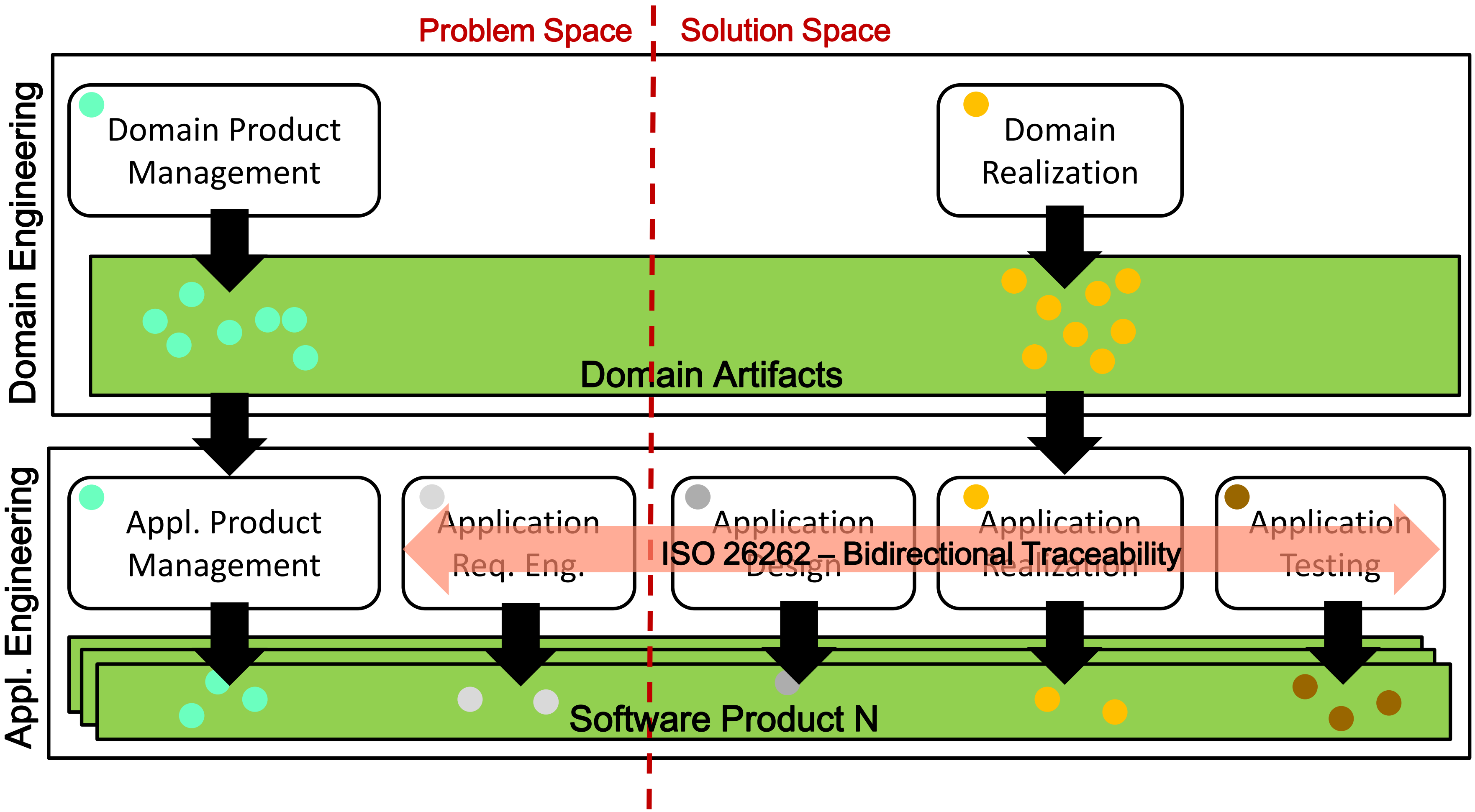
Contents

Introduction to Software Product Line Engineering (SPLE)

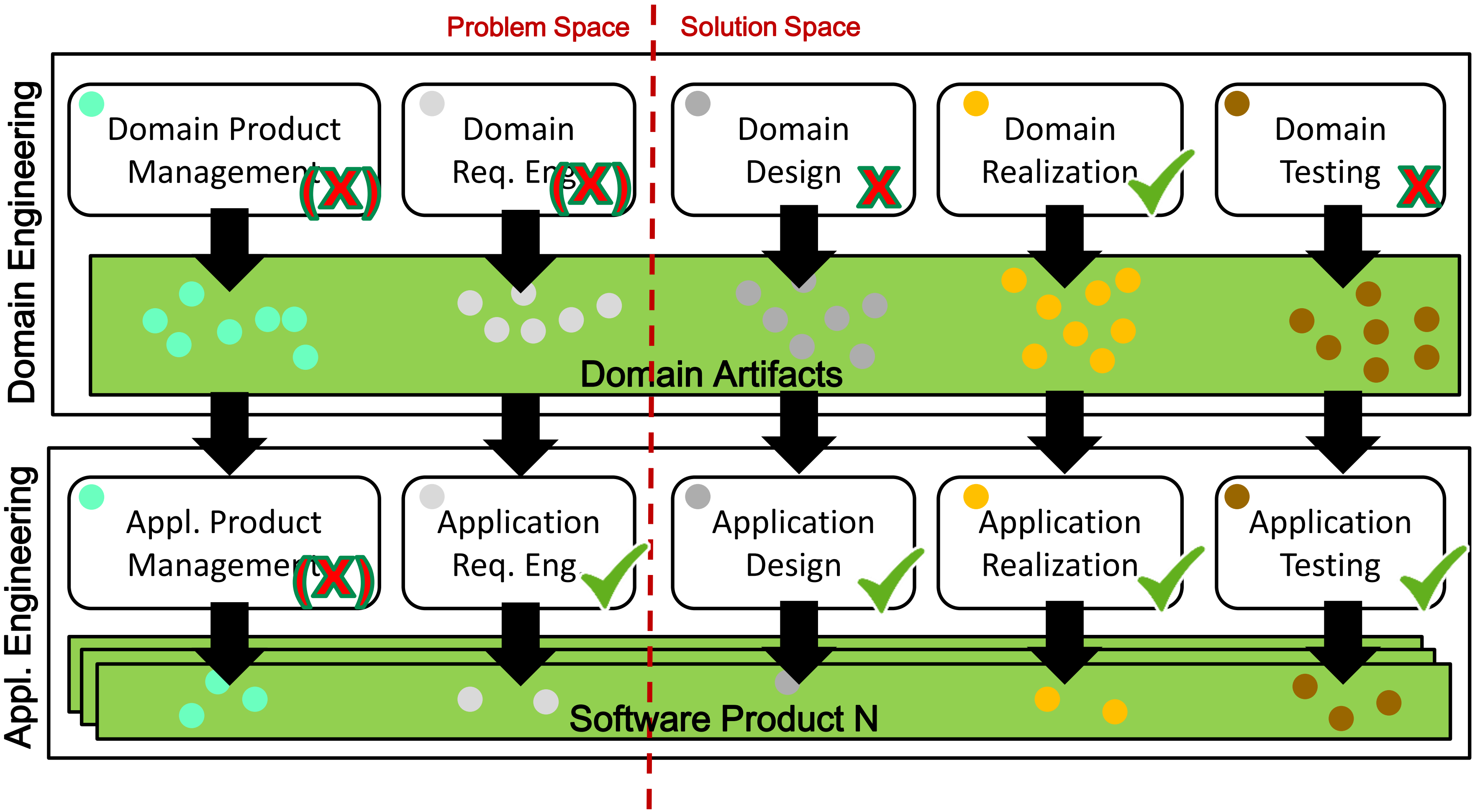
- **Domain Engineering**: Define and realize the commonality and the variability of the software product line.
- **Application Engineering**: Deriving product line applications from the platform.



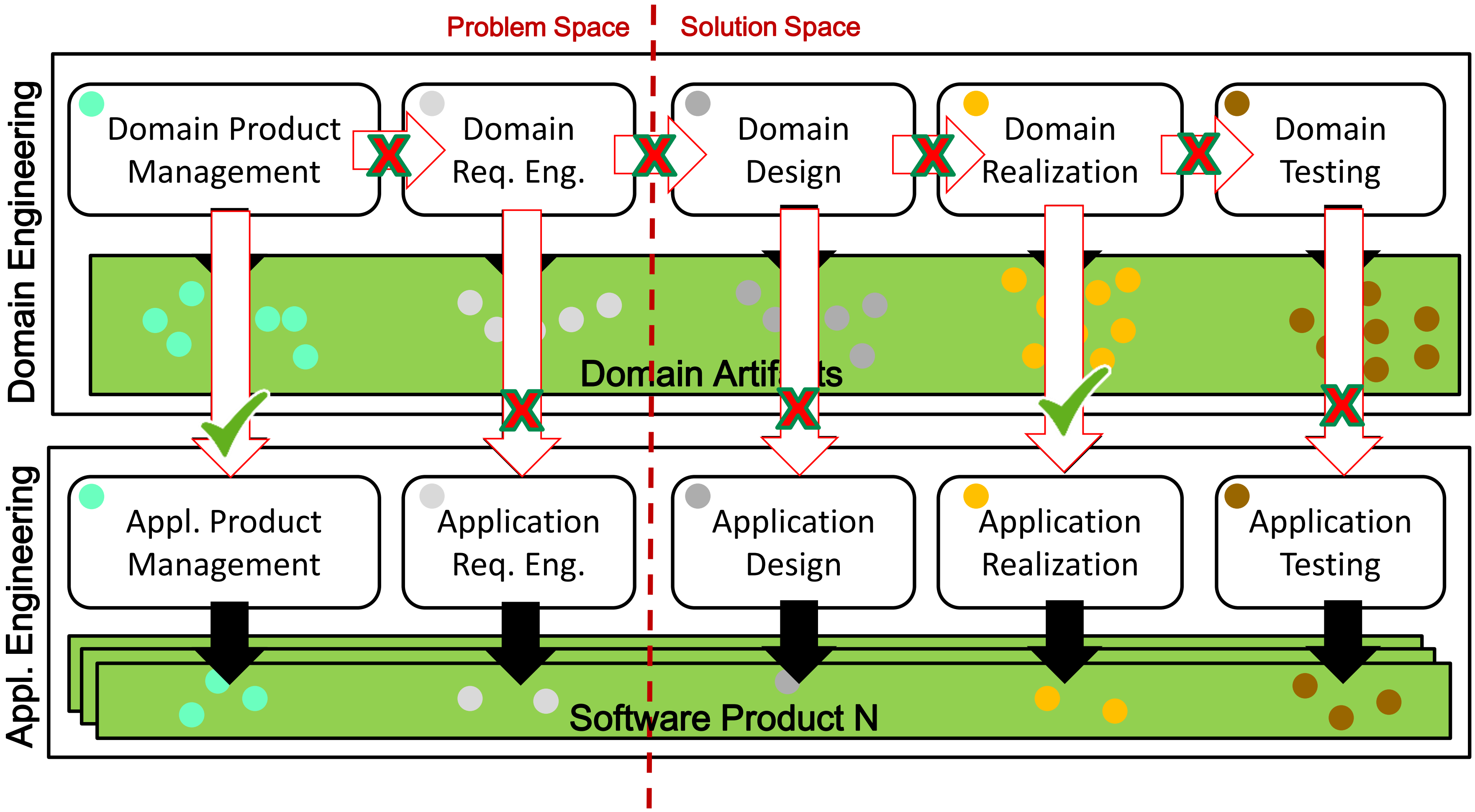
Current Status: Use of SPLE in the Automotive Domain



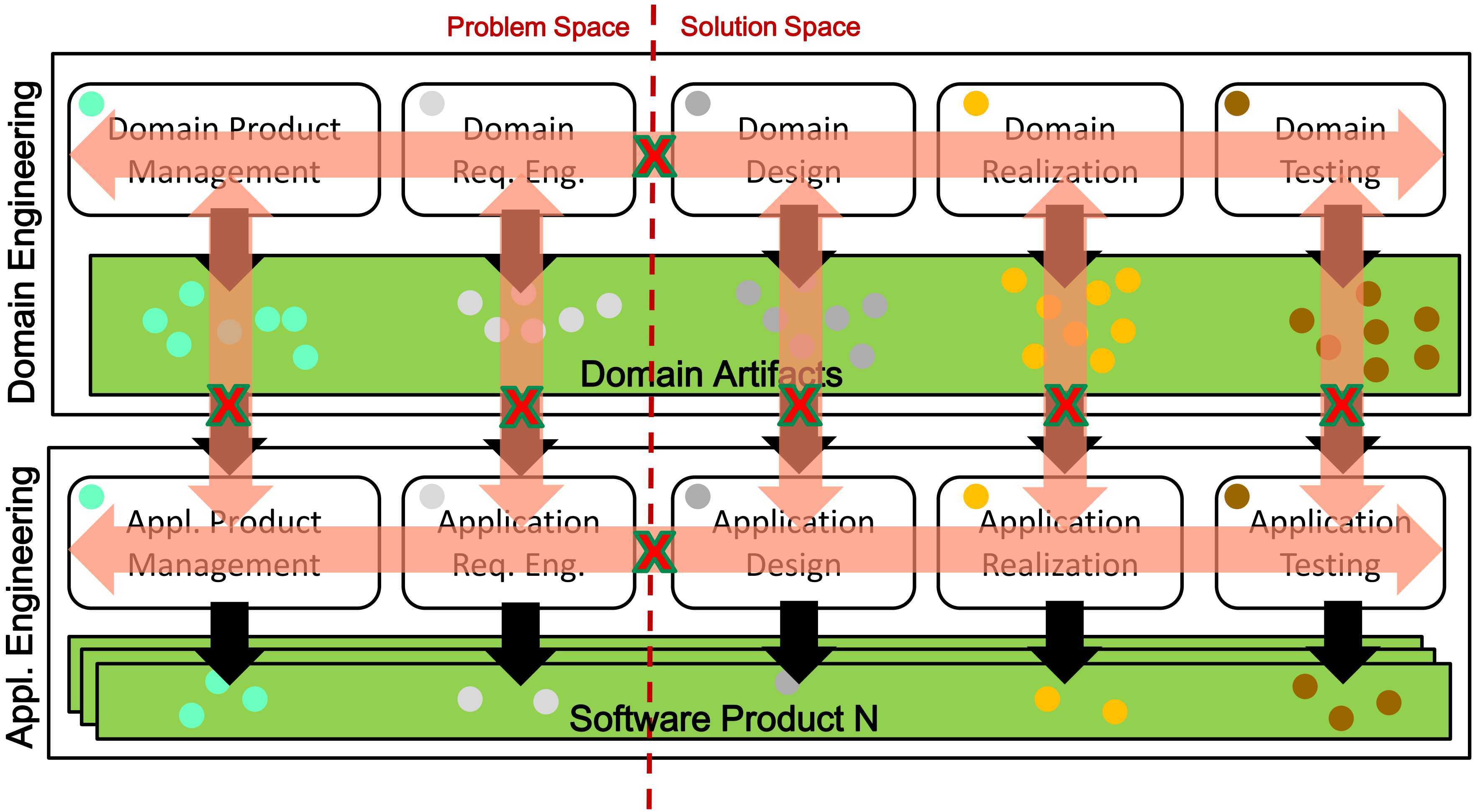
Challenge: Modelling of Requirements and Design Artifacts



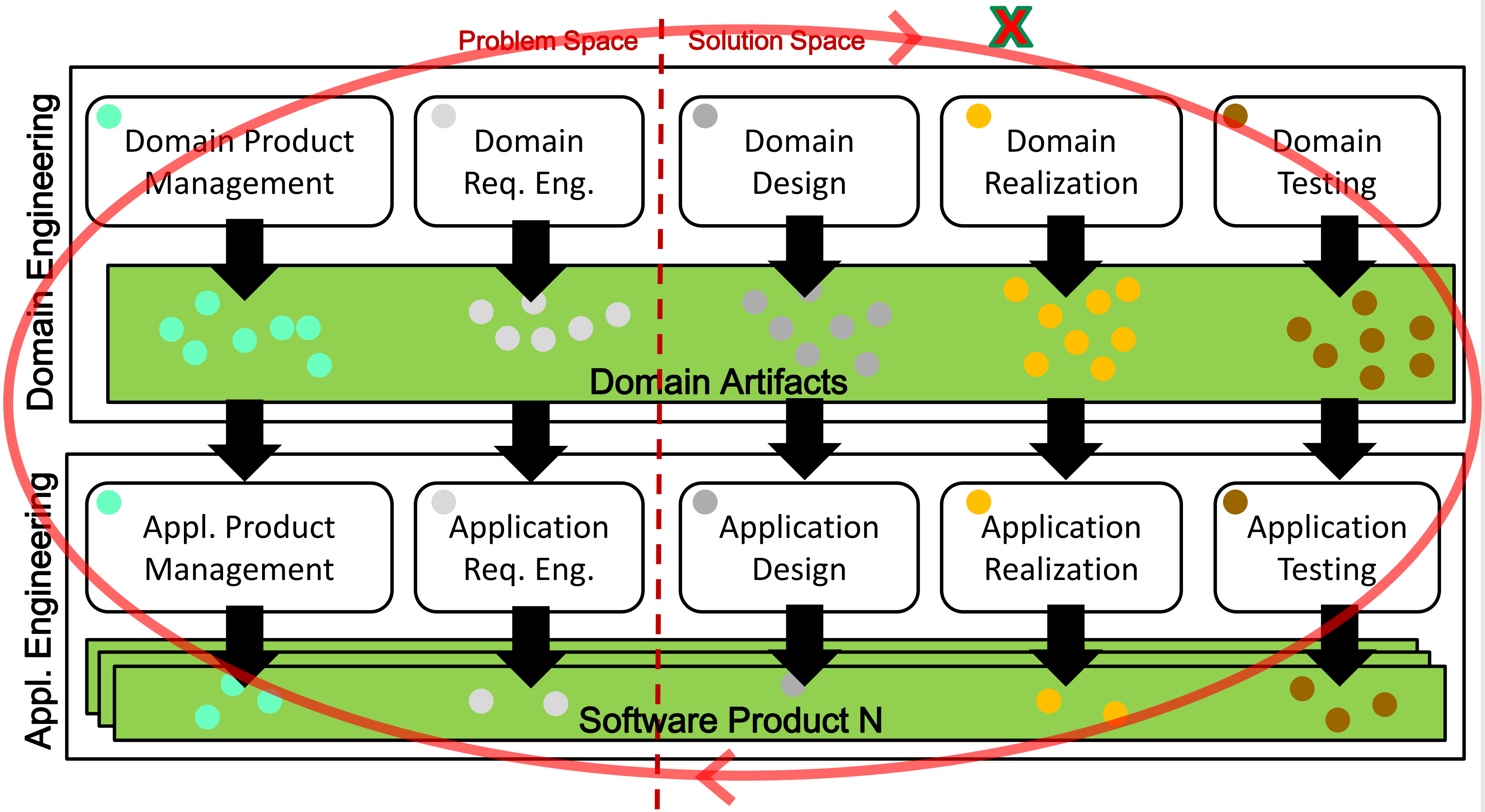
Challenge: Variability and Configurability



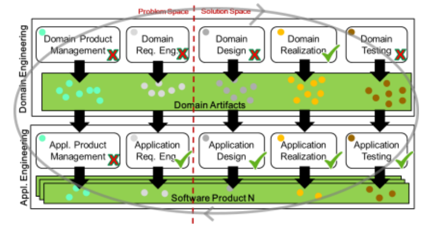
Challenge: Traceability of all Development Artifacts



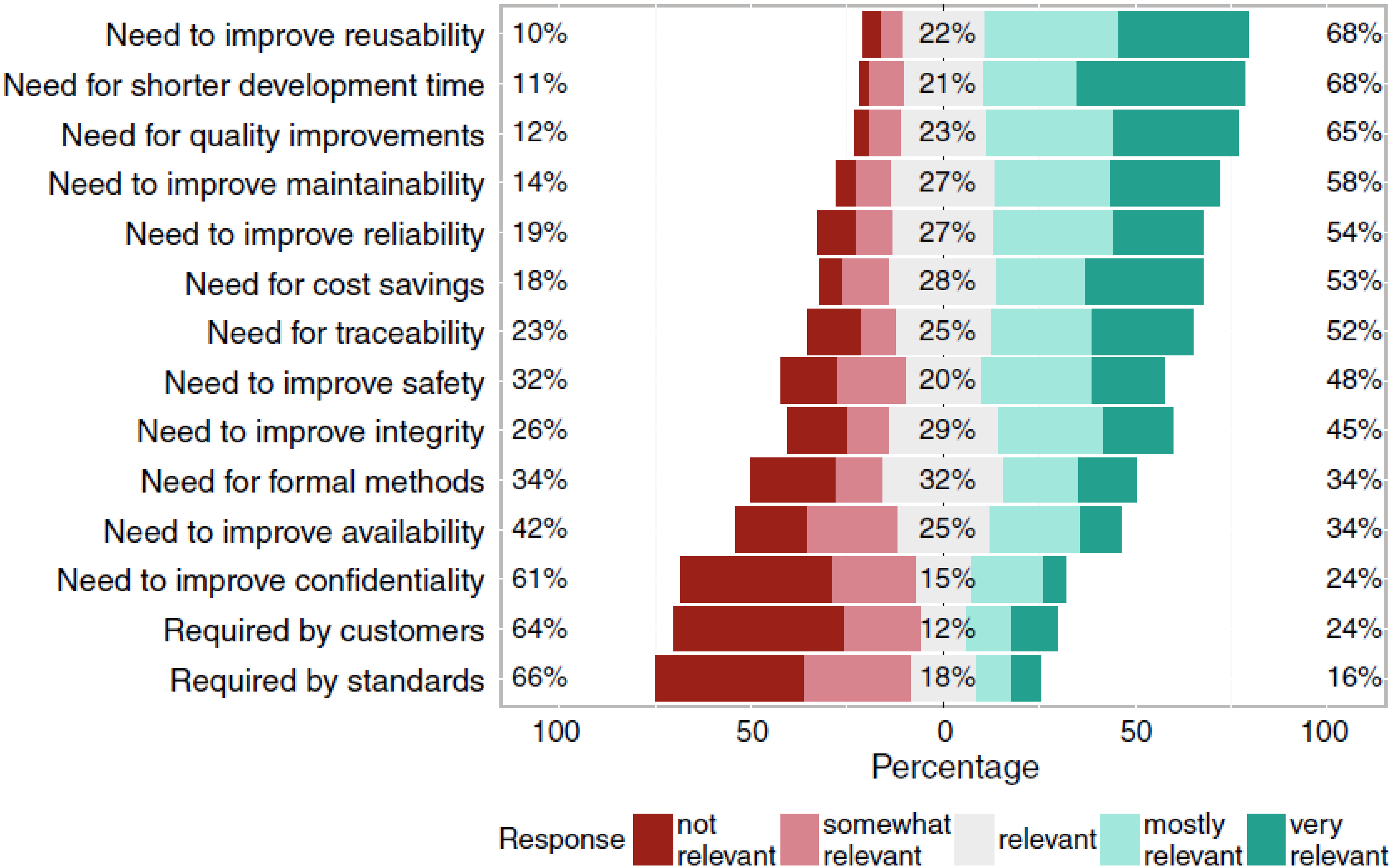
Challenge: Lifecycle Management

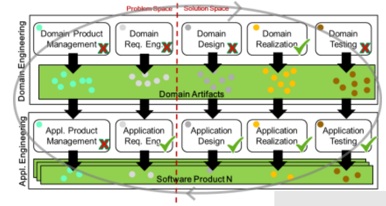


Contents

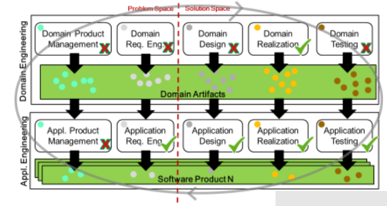

Problem Space
Solution Space
Domain Engineering
Domain Product Management
Domain Req. Eng.
Domain Design
Domain Realization
Domain Testing
Domain Artifacts
Appl. Engineering
Appl. Product Management
Application Req. Eng.
Application Design
Application Realization
Application Testing
Software Product N

Importance of Modelling: Many Reasons for Introducing Model-based Engineering

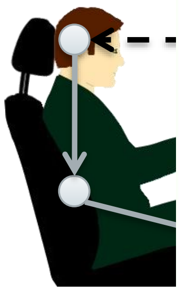


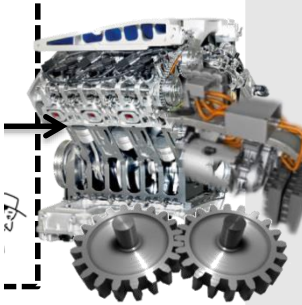


Reference: G. Liebel, N. Marko, M. Tichy, A. Leitner, J. Hansson: Model-based engineering in the embedded systems domain: an industrial survey on the state-of-practice. Software & Systems Modeling, 17(1), 2018, pp. 91-113.



Importance of Architecture: Real World Example “Longitudinal Dynamics Torque Coordination”

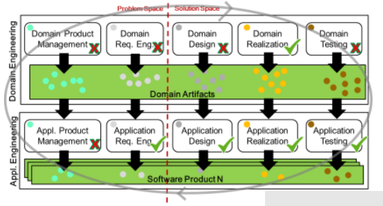




- Unc com betw com
- Mut coo
- Fun repl anot context

Data
odes /
on Information

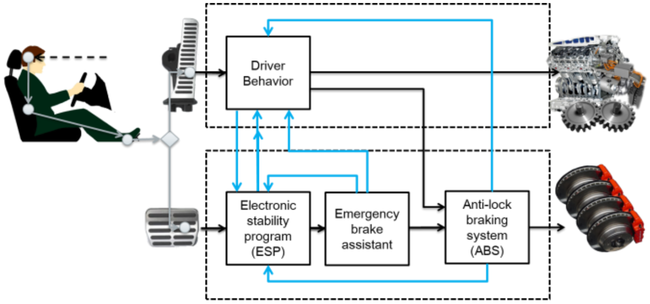
➔ Huge effort of maintenance and further development



Importance of Architecture: Real World Example "Longitudinal Dynamics Torque Coordination"

- Some facts about the power train software system:

Element Type*	Count (in 2015)
Projects (versioned)	6.533
Modules (versioned)	21.734

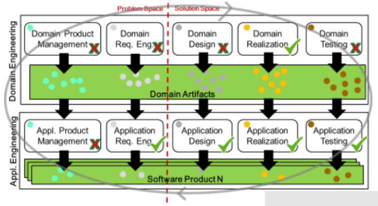


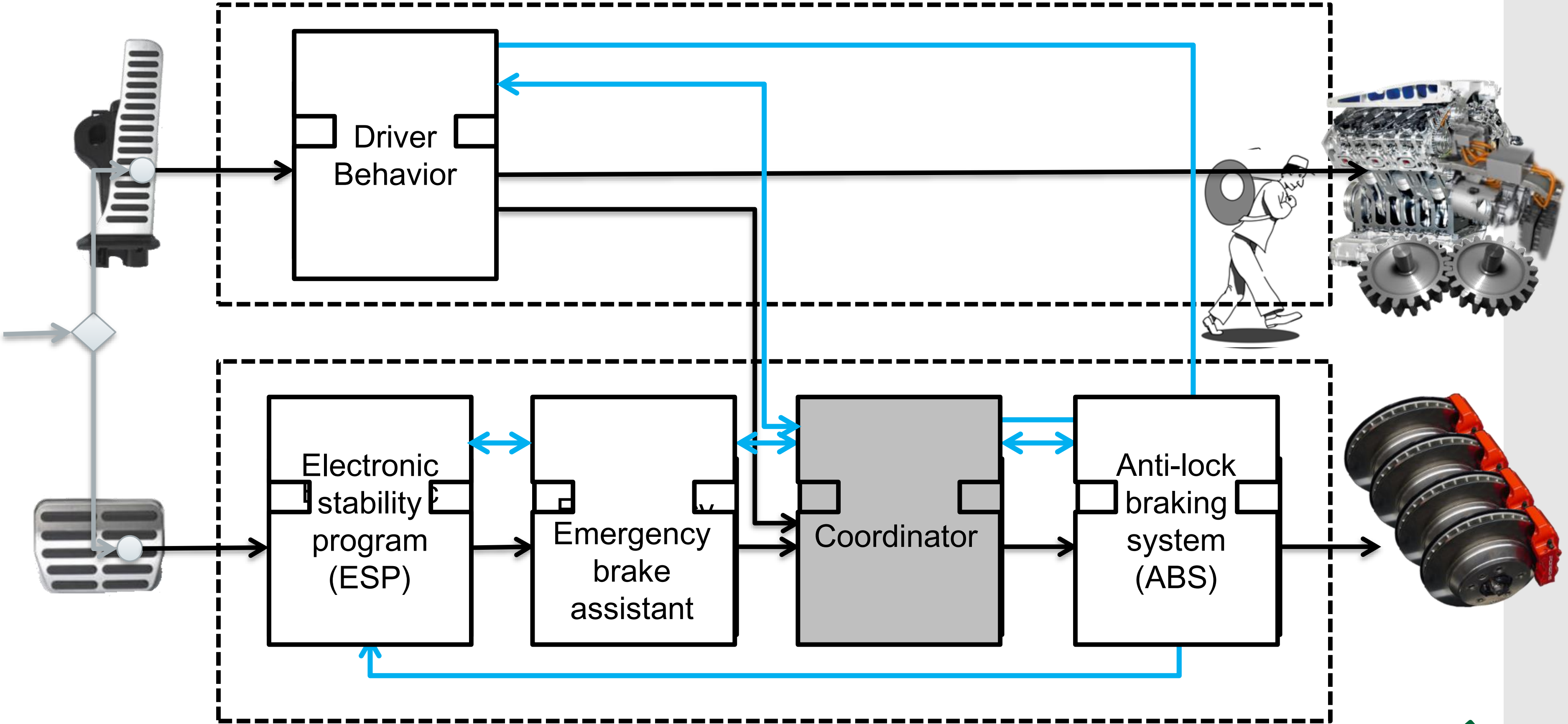
Element Type** (Project Example: MC6)	Count (in 2017)
System constants (SC)	~18.000
Application parameters	~80.000

*Source: ADD

**Source: DCM, CSV

Importance of Architecture: Real World Example
Introducing new Architecture Concepts





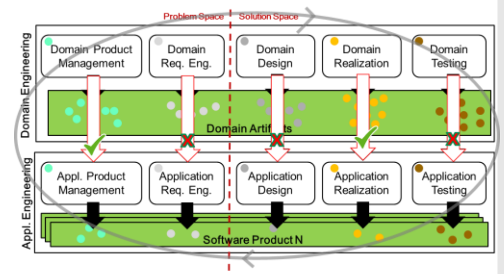
- Separation of concern
- High cohesion
- Loose coupling

→ Reducing the accidental complexity ✓

Contents

1. Initial Situation: Platform Strategy in Automotive Industry

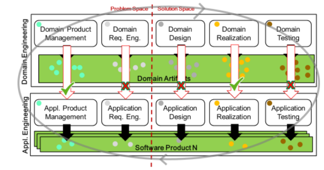
2. **From Software Sharing to Software Product Line Architecture and Product Management – Challenges**
    1. Modelling of Requirements and Design Artifacts
    2. **Variability and Configurability**
    3. Traceability of all Development Artifacts
    4. Lifecycle Management



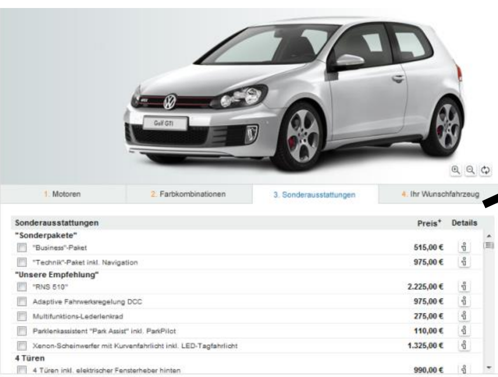
3. Our Approach: Connecting Product Management and Software Product Line Architecture

4. Under the Hood: Concrete Modelling Approach Illustrated by Example

5. How to Cope with the Challenges of the Digital Transformation

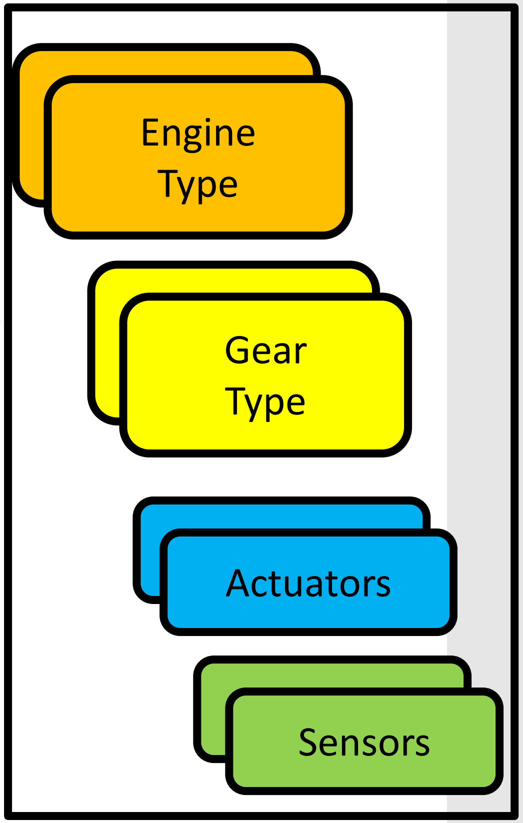


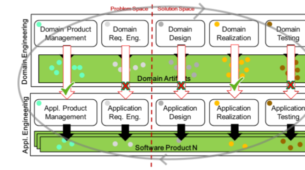
Where It All Begins: Product Definition and Management



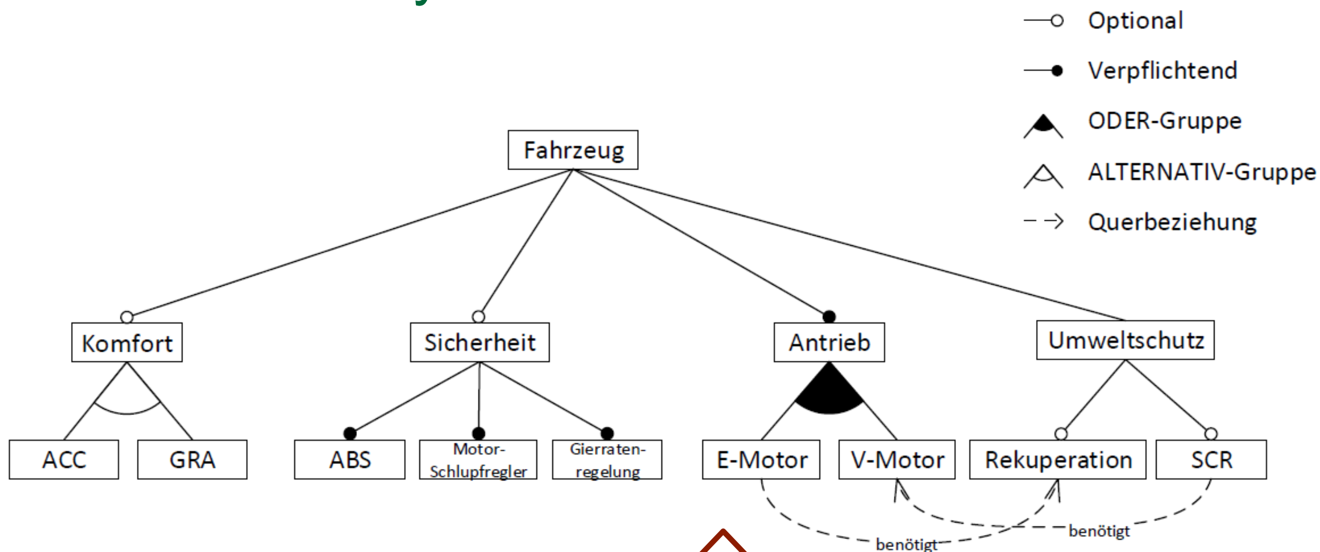
Volkswagen Konfigurator, [http://www.volkswagen.de]

- Trendline
- Sportline
- Comfortline
- R-Line
- …

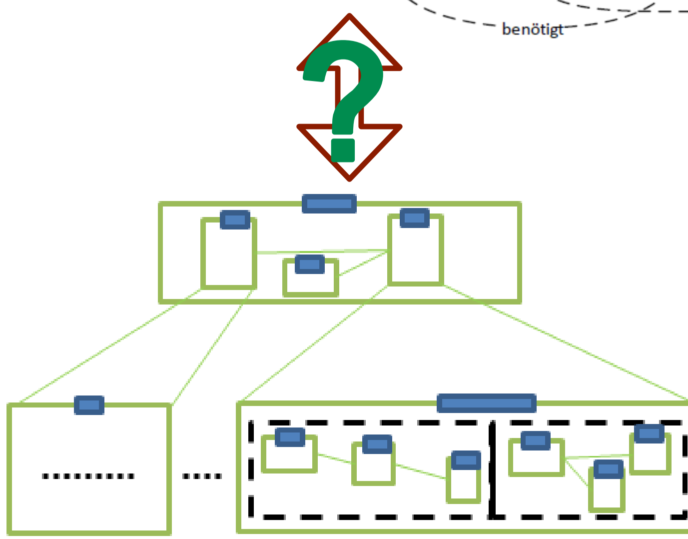




Product Definition by Feature Trees



Feature Tree

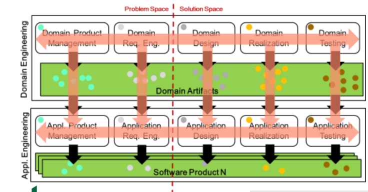


Software Product Line Architecture

Product Definition: Missing Modelling Concept and Impact to Architecture Configuratíon

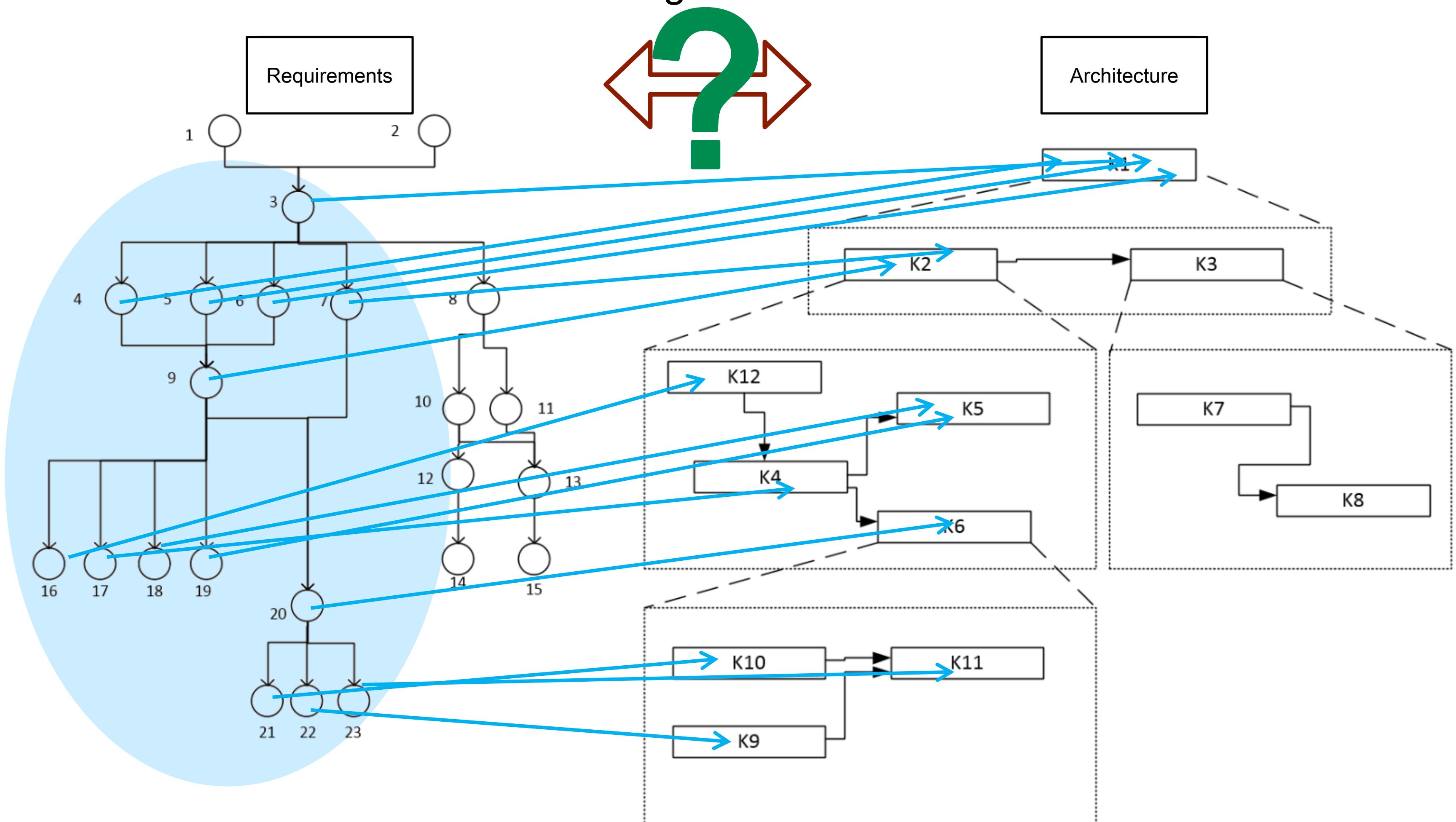
TU Clausthal

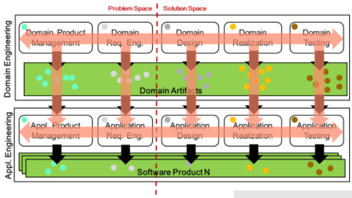
Contents



Enabling Traceability between Requirements and Architecture

- According to ISO 26262: Documentation must be delivered traceable:
  - Effort for maintaining traceability between requirements and architecture
  - Effort for delimitation of changes



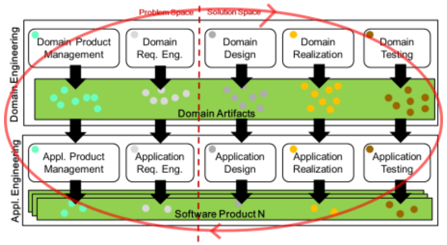


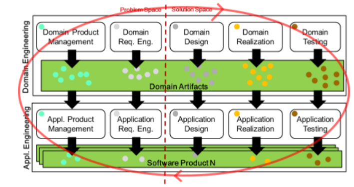
Example: Steering System at Volkswagen

- Initial situation: Development of a release of a new steering system finished
- No traceability between requirements and architecture
- According to ISO 26262: All traces between 8.000 requirements and 200 modules had to be set manually (Ø 1.000 LOC per module)
- Effort: 50 person-years, Costs: 15 Million EUR (incl. costs for 9 month delay of SOP)
- For comparison: Estimated effort, in case the traces had been set before product realization: 15 person-years (1,5 Million EUR)

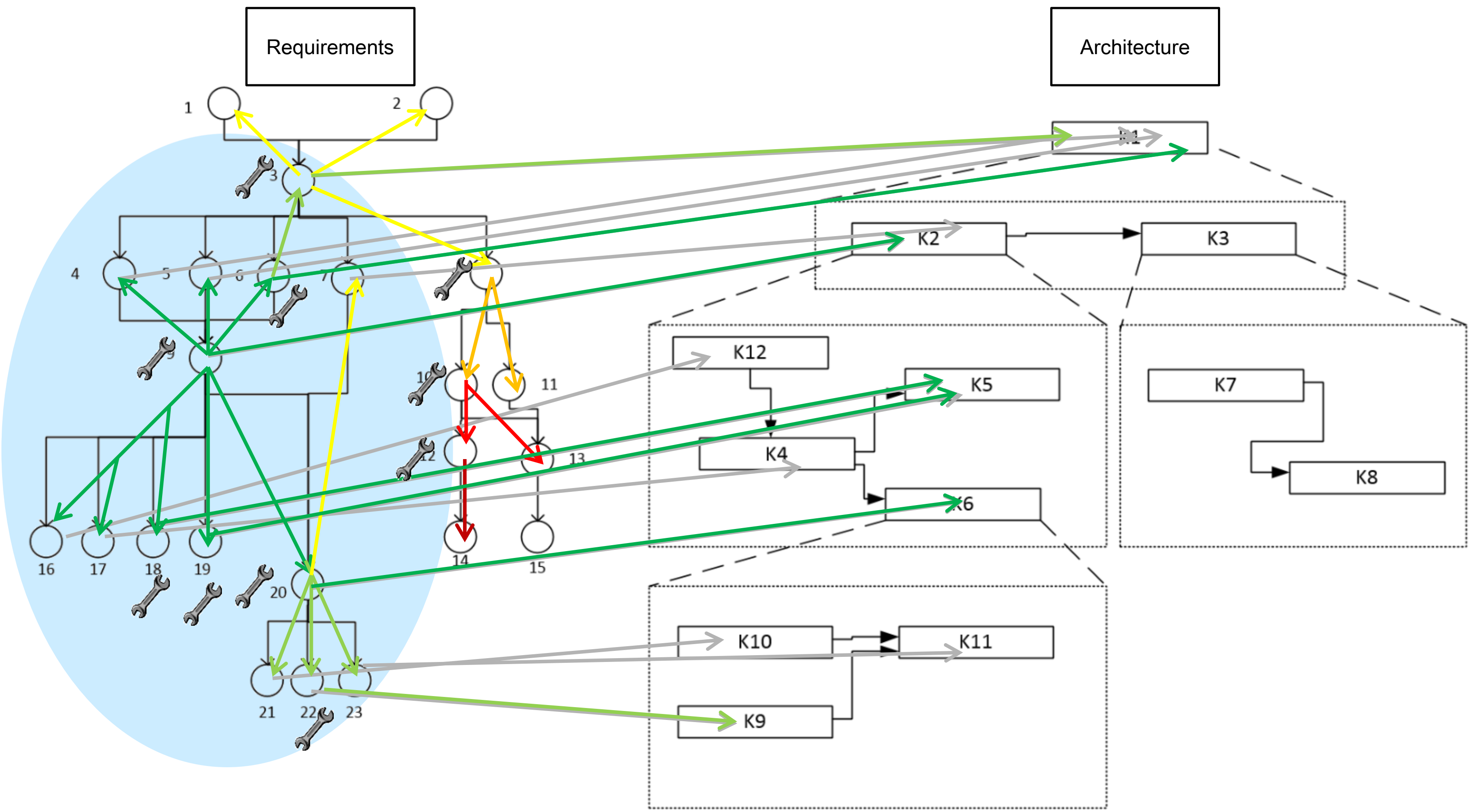
In case of a new product development, traces have to be set again!

Contents

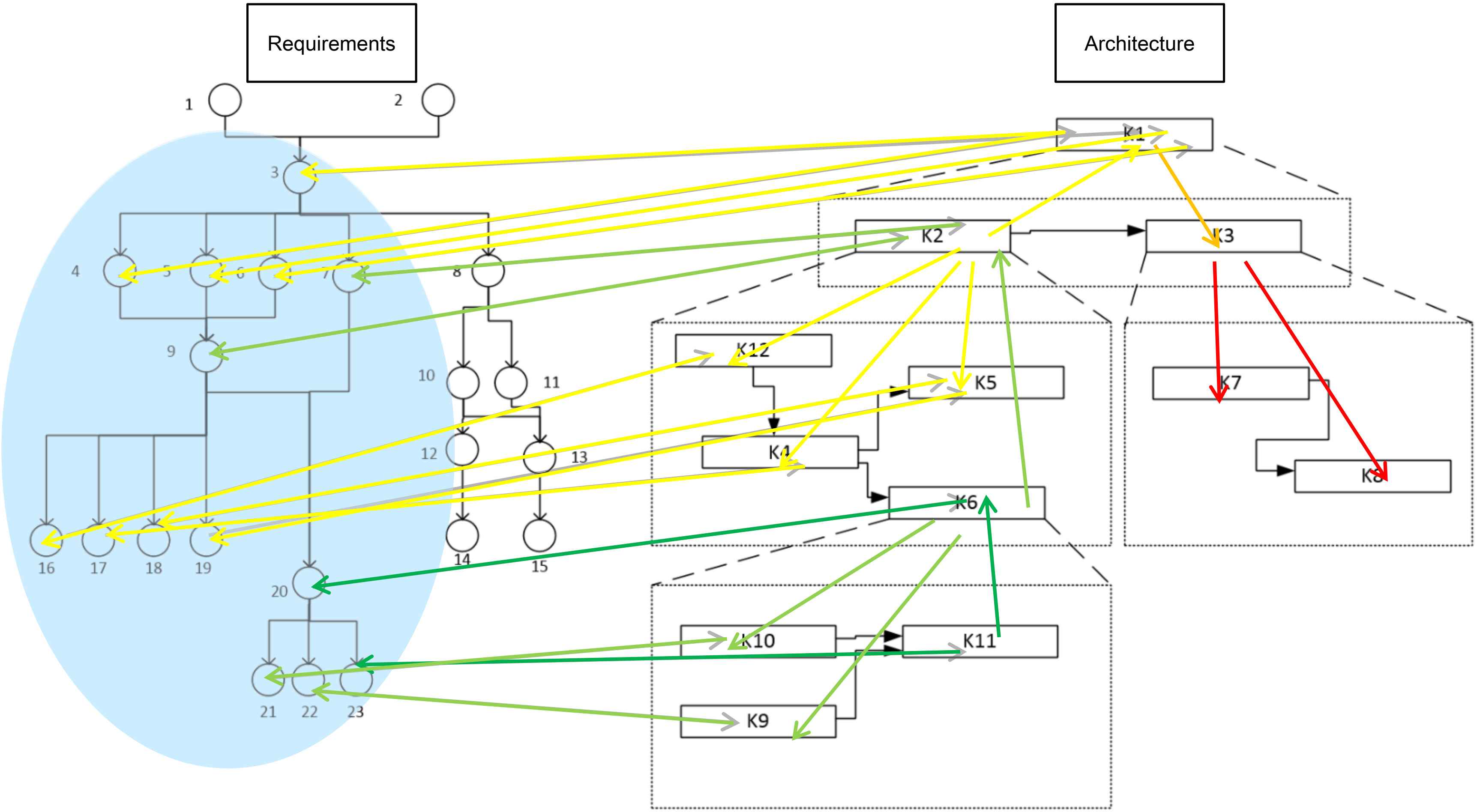

Problem Space
Solution Space
Domain Engineering
Domain Product Management
Domain Req. Eng.
Domain Design
Domain Realization
Domain Testing
Domain Artifacts
Appl. Engineering
Appl. Product Management
Application Req. Eng.
Application Design
Application Realization
Application Testing
Software Product N



Modification to Requirement 9



Modification to Component K11

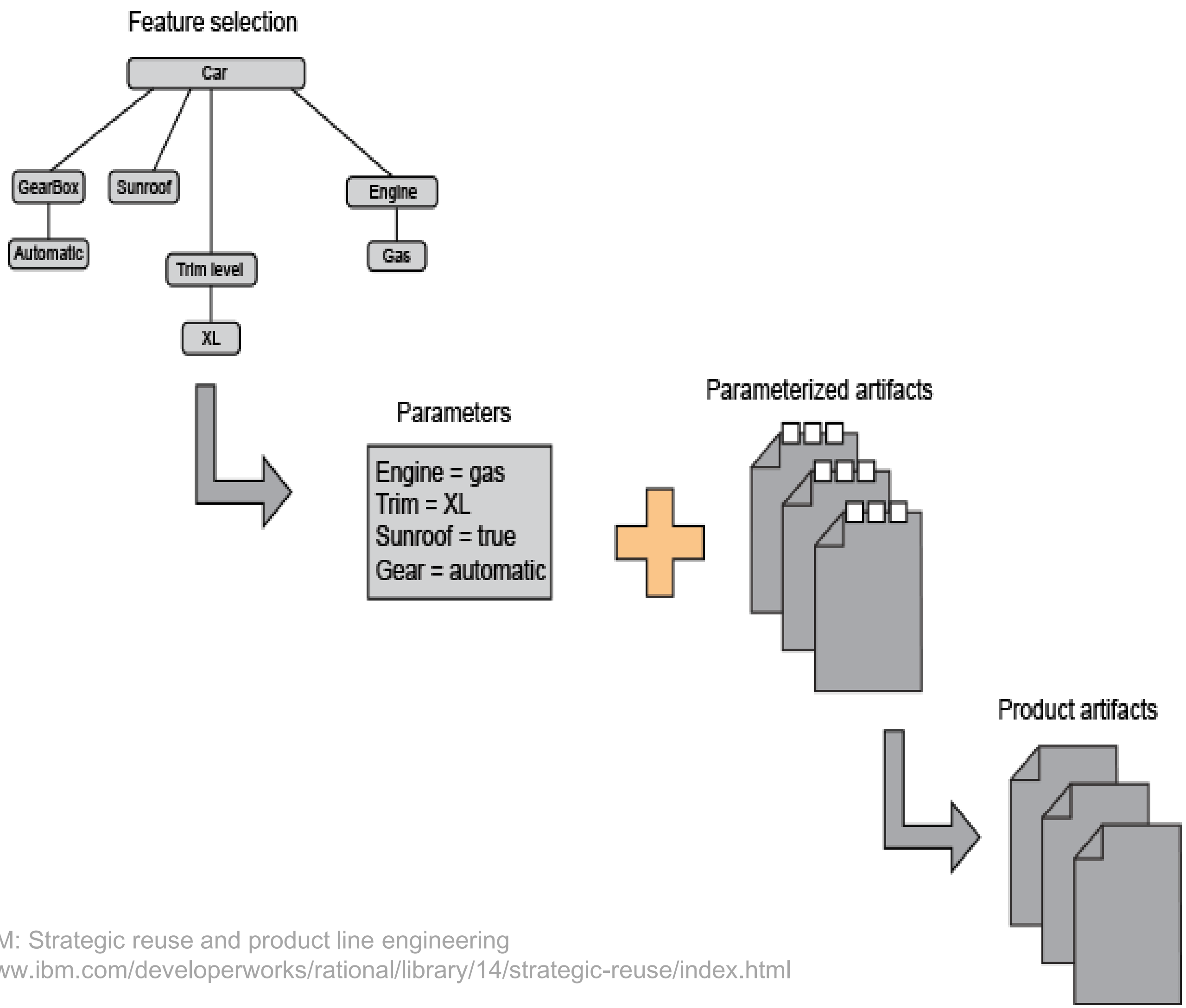


Contents

Challenges in Product Management and Software Product Line Architecture

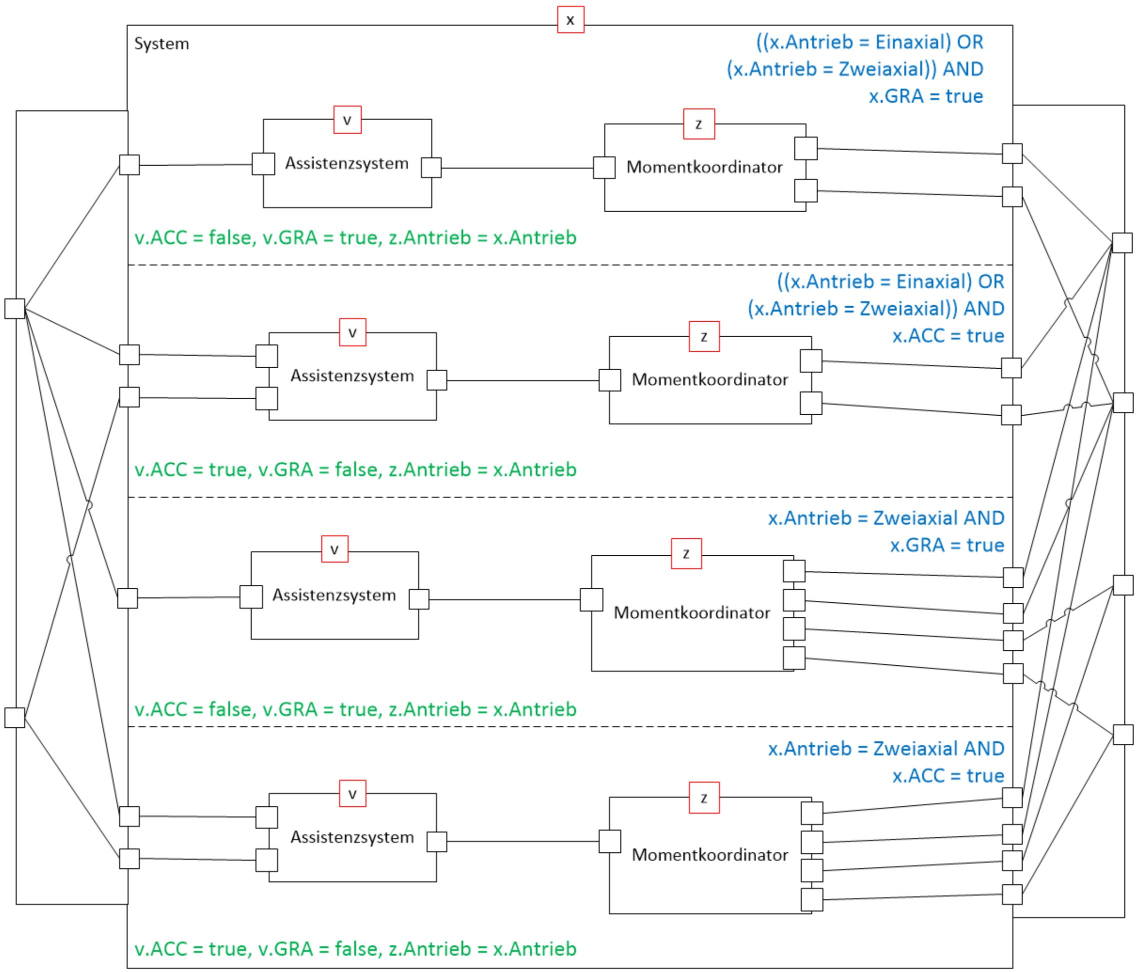
- Scalable modelling approaches supporting variability
  - on requirements level, e.g., feature trees
  - on design level, e.g., SPLA
- Variability and configurability, e.g.,
  - determination of variability decisions propagated on domain level
  - derivation of artifacts on product level
  - derivation of traces on product level
- Providing and maintaining traceability, e.g.,
  - between feature tree and requirements
  - between requirements/features and SPLA
  - between SPLA and code artifacts
- Lifecycle management between domain / application engineering

Approach: Parameterized Artifacts and Propagation of Variability Decisions



Reference: IBM: Strategic reuse and product line engineering
See: https://www.ibm.com/developerworks/rational/library/14/strategic-reuse/index.html

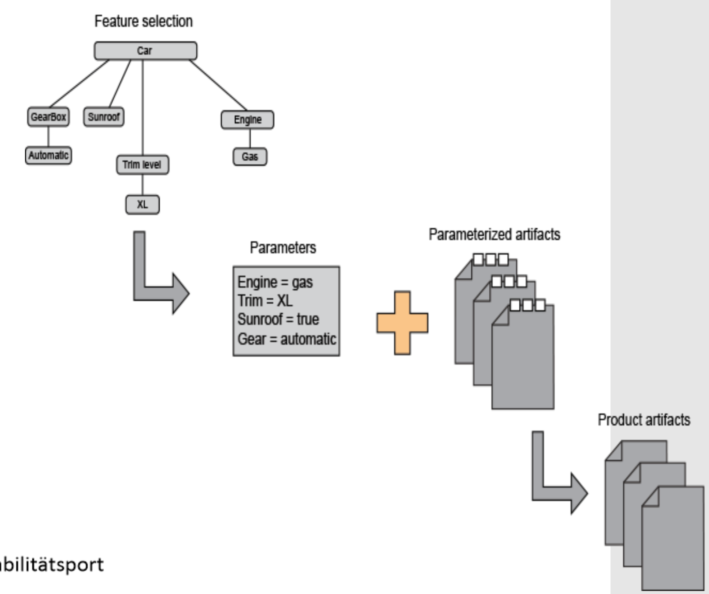
Handling Variability in the Implementation Artifacts



x = {Antrieb, GRA, ACC}

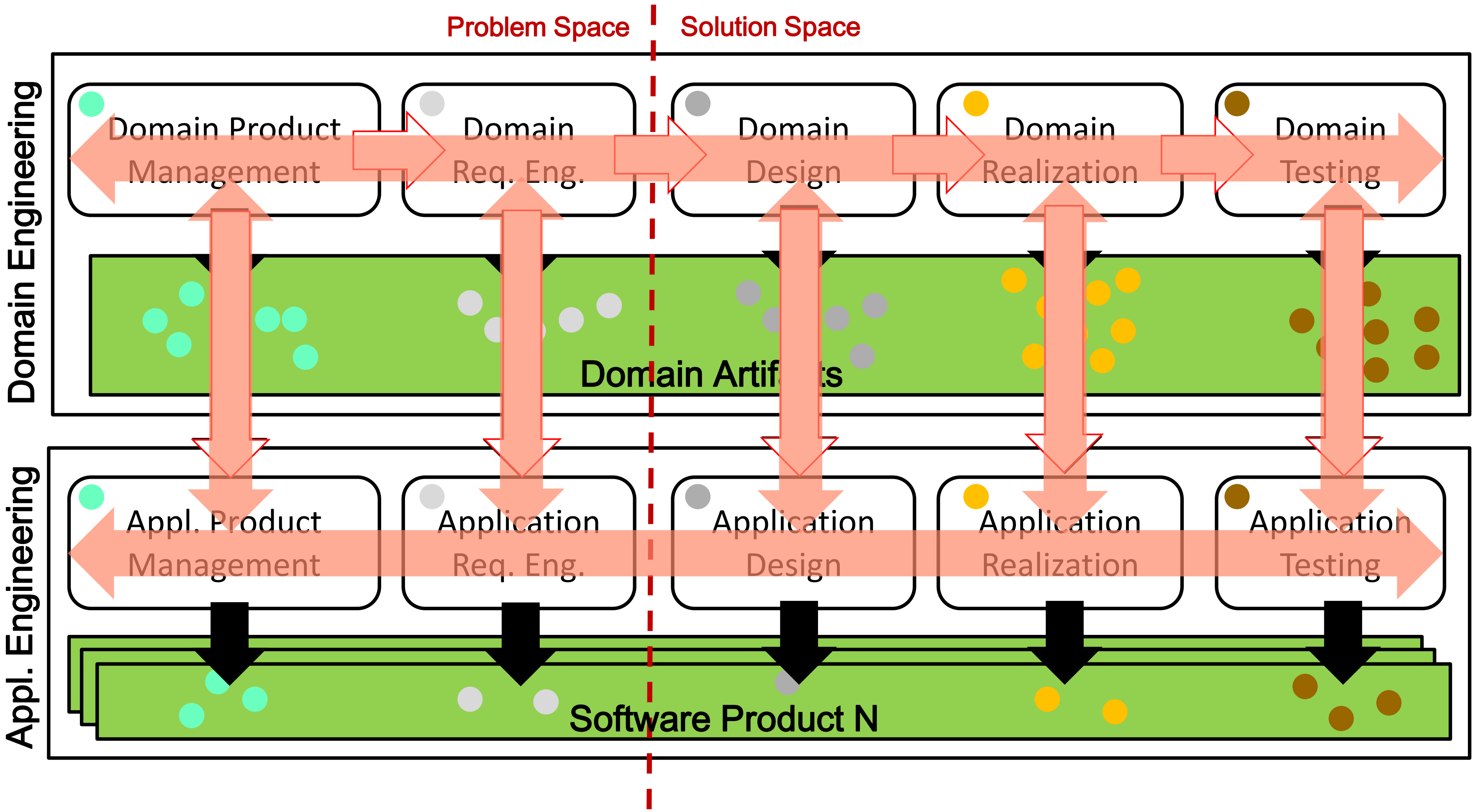
v = {ACC, GRA}

z = {Antrieb, Einzelradantrieb}



	Variabilitätsport
Ausdruck	Auswahlbedingung für Strukturvariante
Ausdruck	Belegung von Variablen in entsprechender Strukturvariante

Approach: Parameterized Artifacts and Propagation of Variability Decisions including Derivation of Traces

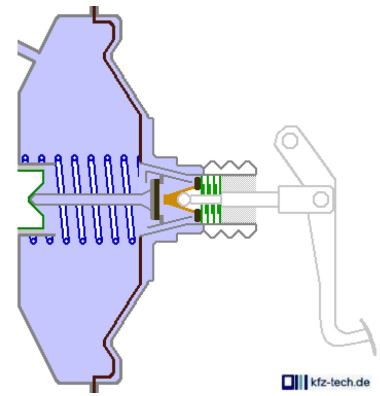


Benefits: Enabling Traceability + Controlling Variability + Managing Evolution

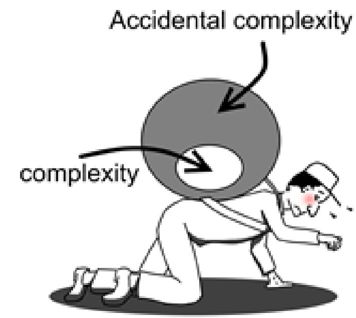
Contents

Importance of Modelling: Real World Example BSU – “Brake Servo Unit Software System”

- Main task: Ensure a sufficient vacuum within the brake booster that is needed to amplify the driver’s braking force
  - Intake manifold as vacuum generator,
  - or electrically / mechanically driven vacuum pump

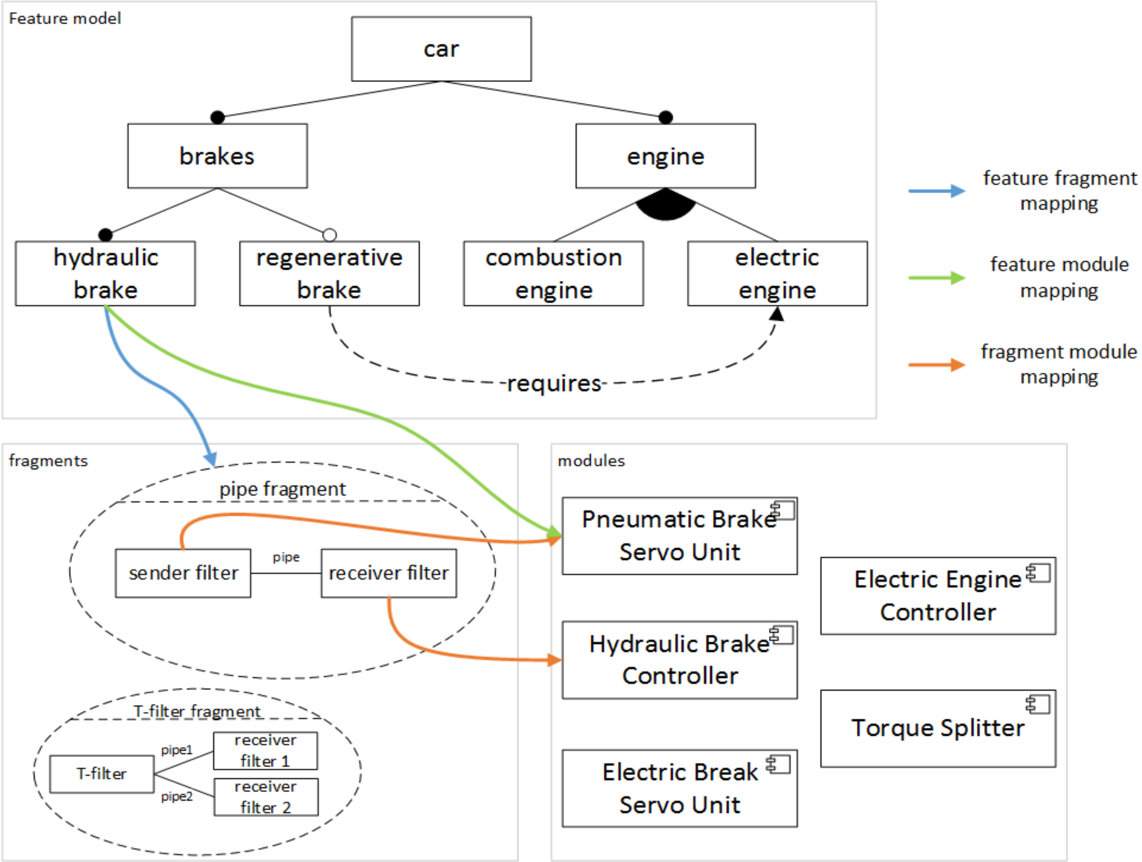


- Before 2012: Implemented by various suppliers, no modeling of architecture
  - Monolithic structure; variability realized completely by annotations; strong coupling and a low cohesion; …
  - → High accidental complexity



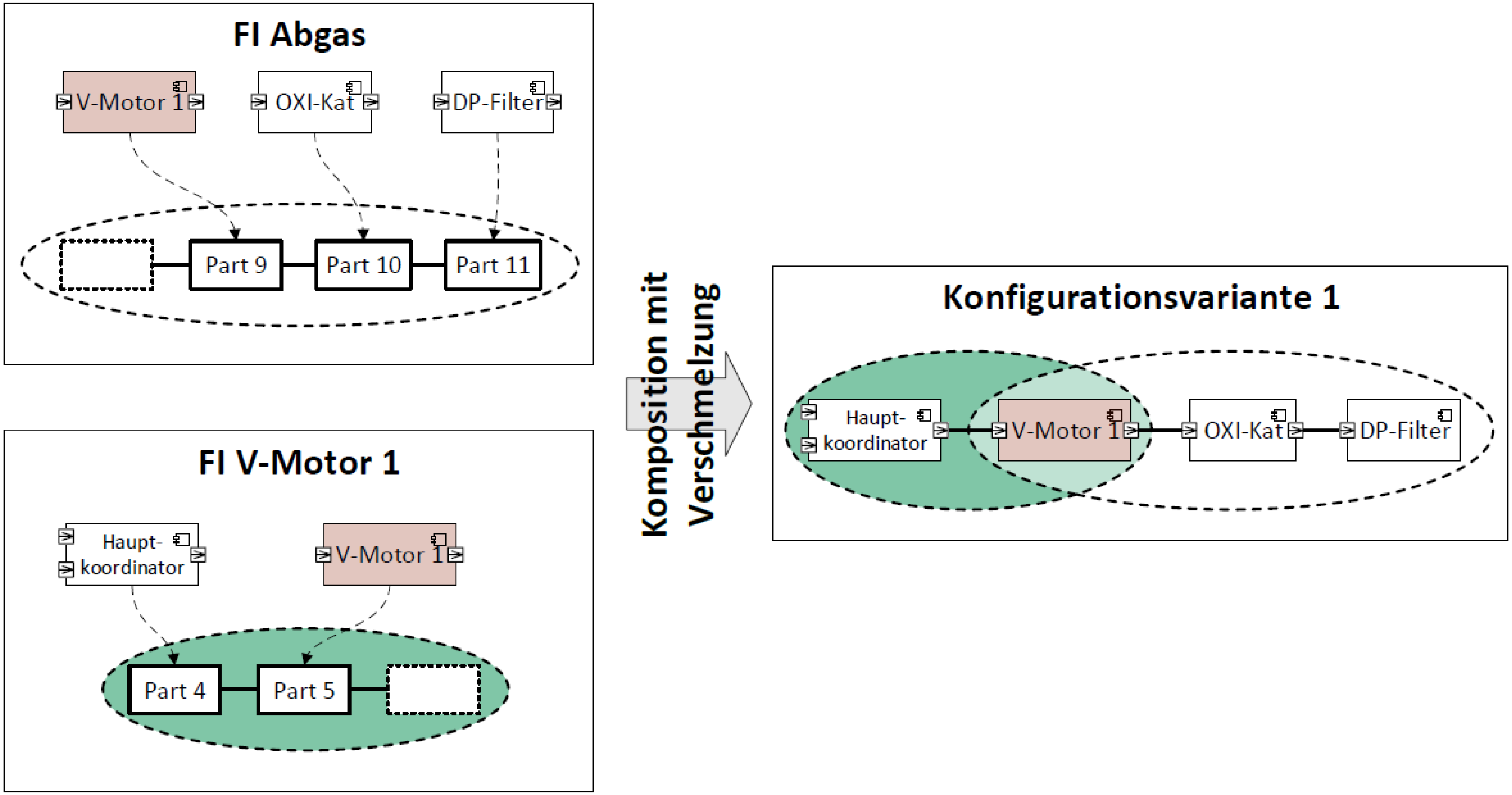
Fragment-based Software Architectures for Product Lines

- Objective: Linking between features and components
- Challenge: Assignment of feature to component not unique
- Fragment-based solution approach: Automated generation of component-based architectures from feature-oriented product lines



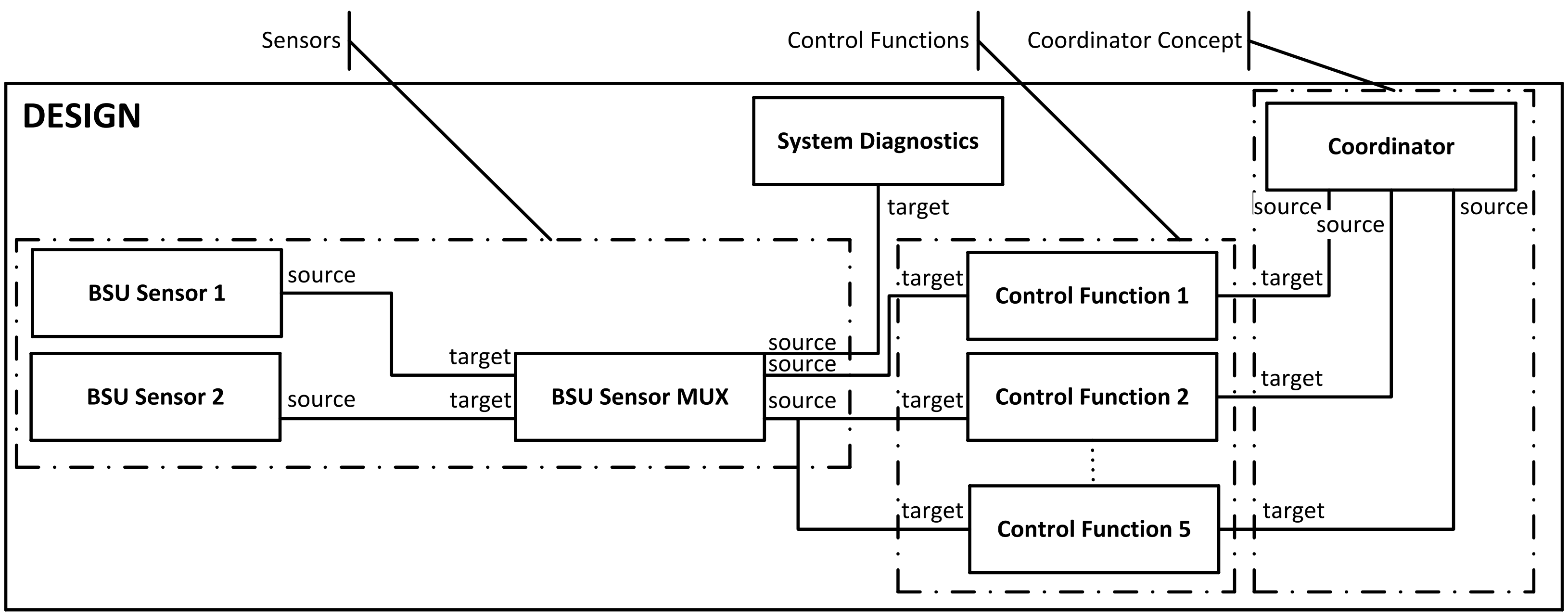
Fragment-based Software Architectures for Product Lines

- Example: The feature implementations of a group are merged into one architecture.



Importance of Modelling: Real World Example BSU

- **In 2012**: New in-house development by Volkswagen and IPSSE
  - Implementation on the basis of the documentation of the existing systems
  - Quality targets: Configurability, extensibility and comprehensibility
  - → Modelling of new architecture and design concepts



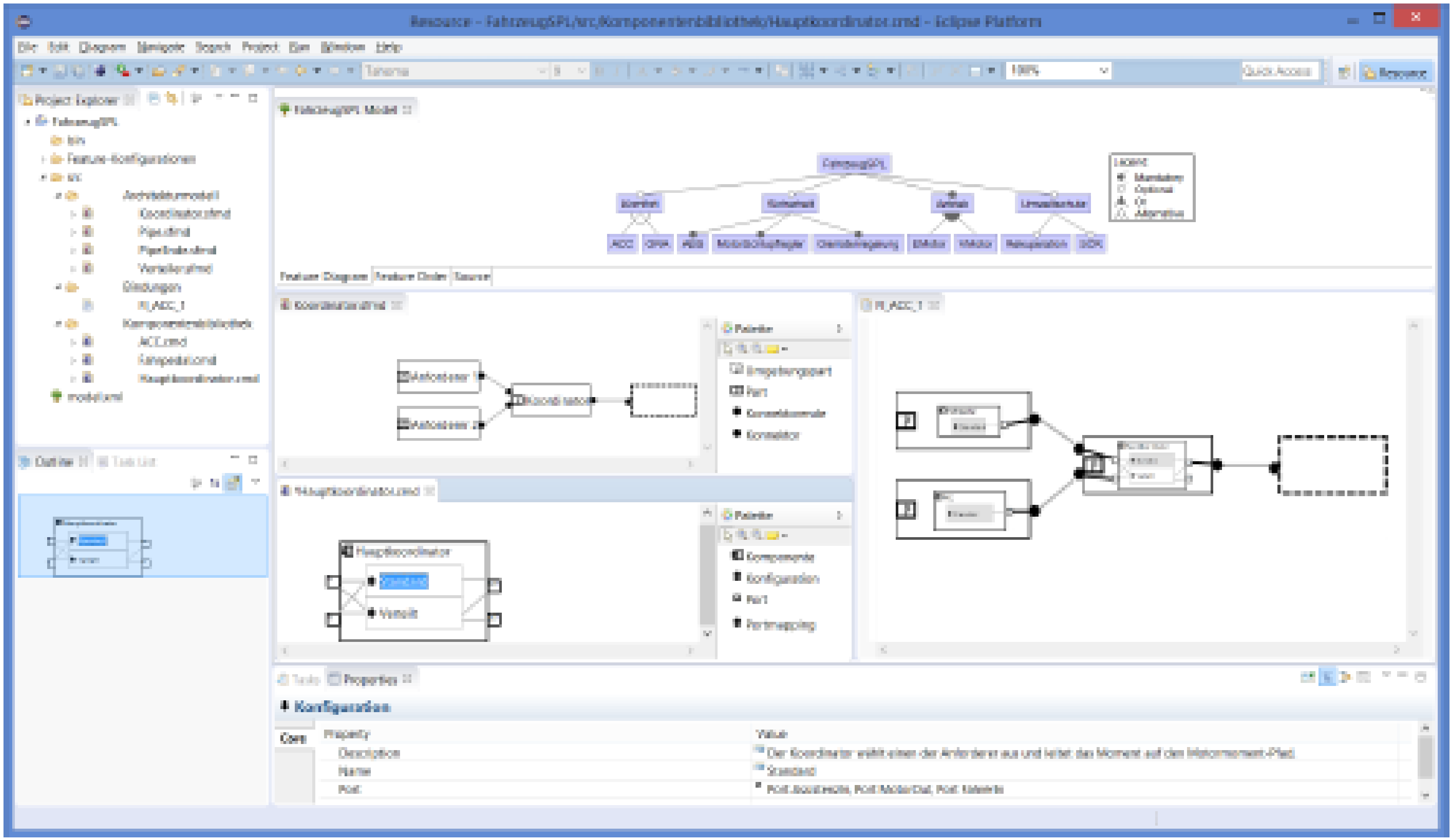
Importance of Modelling: Real World Example BSU
Results of the Case Study (2012-2016)

	Count in 2012	Count in 2016	Number of versions	Average number of versions
Logical architecture elements	10	15	15	1
Module architecture elements	10	15	58	4
Projects	1	21	146	7

	Number of versions used in projects	Cumulated number of versions used over all project versions	Average degree of reuse of each version	Number of used design configurations
Module architecture elements	46	1611	35	n/a
Projects	n/a	n/a	n/a	14

➔ Stable Architecture, High degree of reuse, High number of variants managed!

Effective Tool Support is Essential!



An Experiment for the Configuration Mapping in Prolog...

- 400 lines of Prolog program
- 13 atomic features and feature configuration
- 15 atomic modules with more than 100 variability parameters
- Selection of hierarchical components via solution variables solution alternatives
- ➔ First attempt: Works and parameters were reduced by a factor of 10 (higher factors likely due to cross effects)

```
%Beispiel einer Konfiguration der Konfigurationsvariablen,
%welche nur die Komponenten nutzt, die nicht auf die
Drezahlsteuerung
%und die Drosselklappe abziehlen
/*featureVariable(ac):-false.
featureVariable(ottoMotor):-false.
featureVariable(dieselMotor).
featureVariable(hydraulischerBKV).
featureVariable(elektrischerBKV):-false.
featureVariable(elektrischeUnterdruckpumpe).
featureVariable(hydraulischeUnterdruckpumpe):-false.
featureVariable(esp).
featureVariable(eHeizung).
featureVariable(startStop).
featureVariable(katalysator).
featureVariable(sailAvailable).
featureVariable(drosselklappenSteuerung):-false.*/
```

```
% Die Menge der Konfigruationsvariablen, welche fÃ¼r die
% VariabilitÃ¤t in der Bedingungslogik genutzt wird.
konfigurationVariable(v1, bsu_bCtlFctAcDeac_C_VW, ak, _).
konfigurationVariable(v2, bsu_idxCtlFctAcMst_C_VW, ak, _).
konfigurationVariable(v3, bsu_idxCtlFctAc_C_VW, ak, _).
konfigurationVariable(v4, bsu_idxCtlFctAcTyp_C_VW, ak, _).
konfigurationVariable(v5, bsu_swiConfAlt_C_VW, ak, _).
konfigurationVariable(v6, bsu_idxCtlFctAltMst_C_VW, ak, _).
konfigurationVariable(v7, bsu_idxCtlFctAlt_C_VW, ak, _).
konfigurationVariable(v8, bsu_idxCtlFctAltTyp_C_VW, ak, _).
konfigurationVariable(v9, bsu_bCtlFctElPmpDeac_C_VW, ak, _)
konfigurationVariable(v10, bsu_idxCtlFctElPmp_C_VW, ak, _).
konfigurationVariable(v11, bsu_bEngOpmDeac_C_VW, ak, _).
```

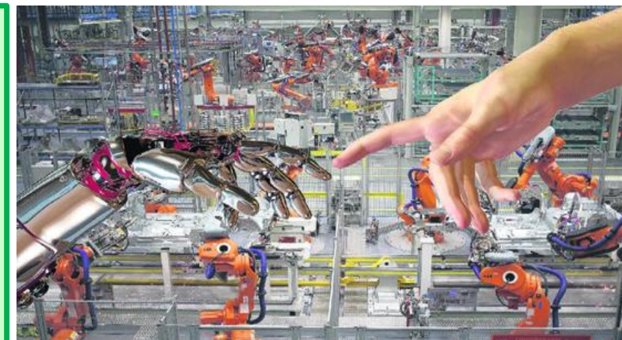
```
Hierarchische Komponenten entstehen durch einen Verbund von
Abstrakten Komponenten. Dabei kann die Auswahl von abstrakten
Komponenten an
Bedingungen geknÃ¼pft sein.
*/
%constraint definition druckkontrolle.
hierarchischeKomponente(hk1, druckkontrolle).
include(hk1, ak15).
include(hk1, ak14).
include(hk1, ak13) :- featureVariable(hydraulischerBKV).
include(hk1, ak12) :- featureVariable(esp).
```

Contents

Future Innovation: Digitization in Almost all Areas at a Rapid Pace

...the innovations of today: Industry 4.0

> “Digitization encompasses the economy and society in its entirety and in all sectors. Therefore, in the opinion of the expert commission, focusing R&I policy on the production sector is counterproductive.“
> (EFI 2016, p.63)



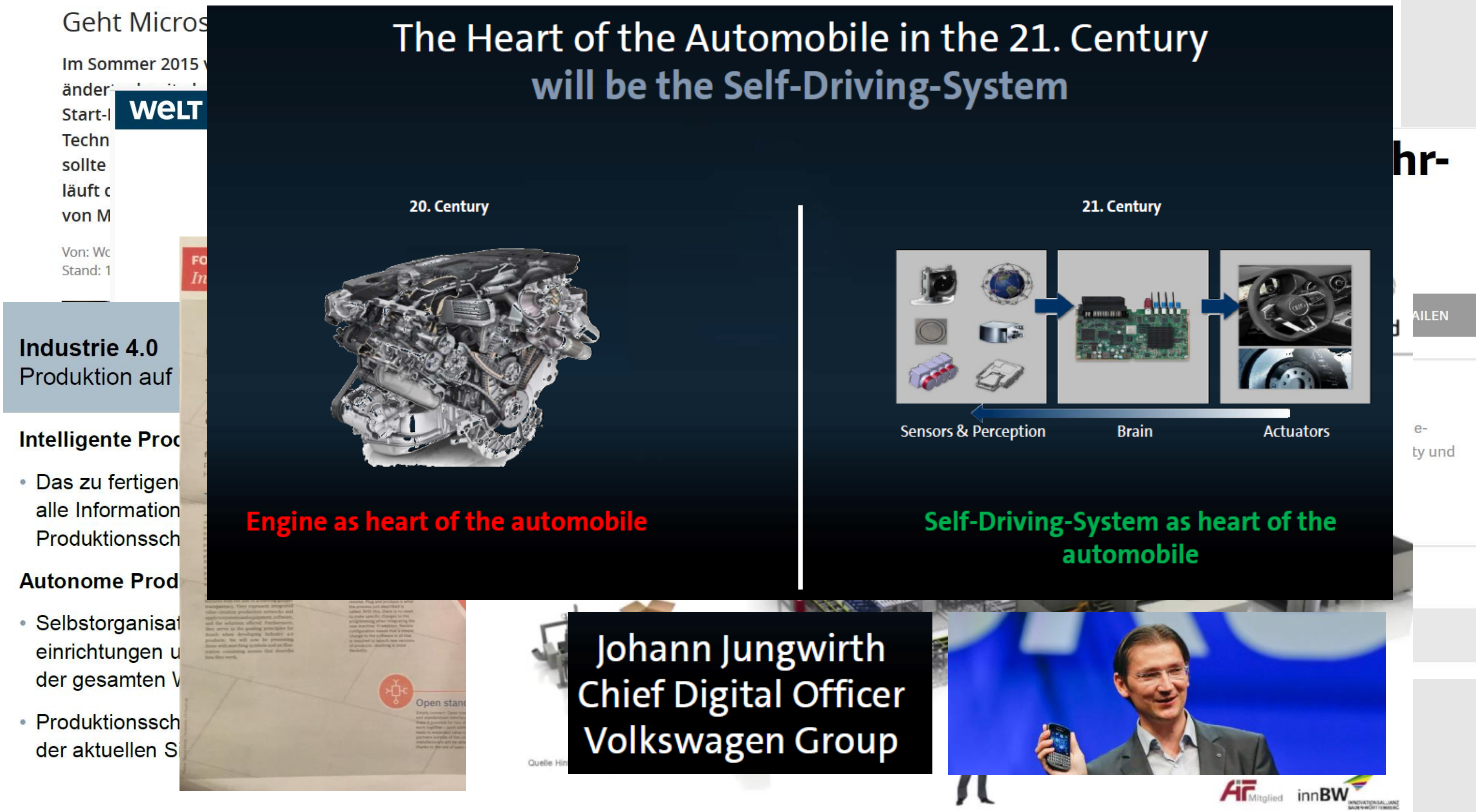
...future innovations: **Digitization - everywhere, rapid and disruptive**

- **Digital ecosystems**: Cloud-based platforms, SoS, ...
- **Transformation to digital agile organization**: Organization with digital genes, products and services
- **Disruptive business models**: Service and data-based
- **Digitization technologies**: AI, big data, connectivity, language assistants, AR, smart sensors, ...
- **Reliability and acceptance**: Safety, security and privacy
- **Digitization skills**: HR, education, …

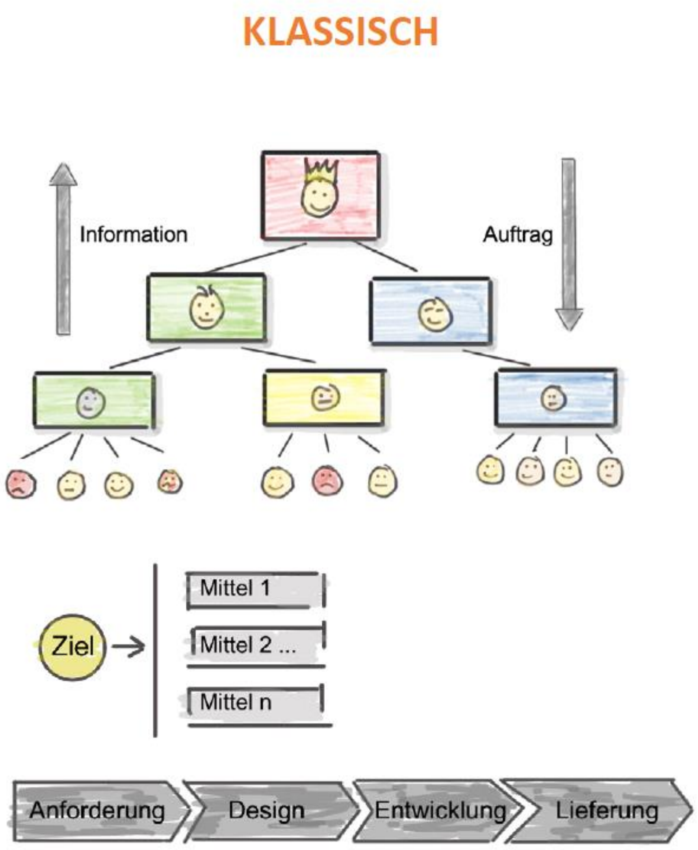
Google

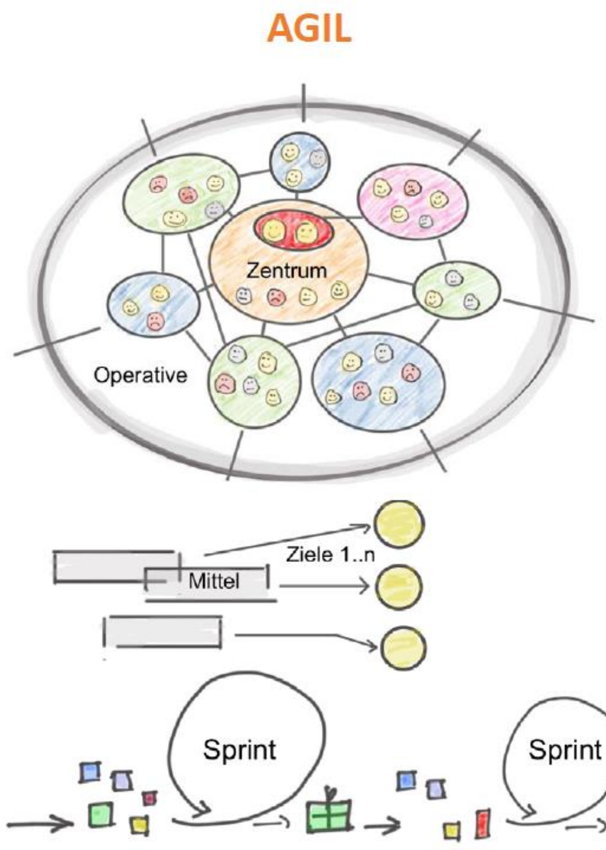
facebook

Digital Ecosystems - The Battle for the Platform is in Full Swing!



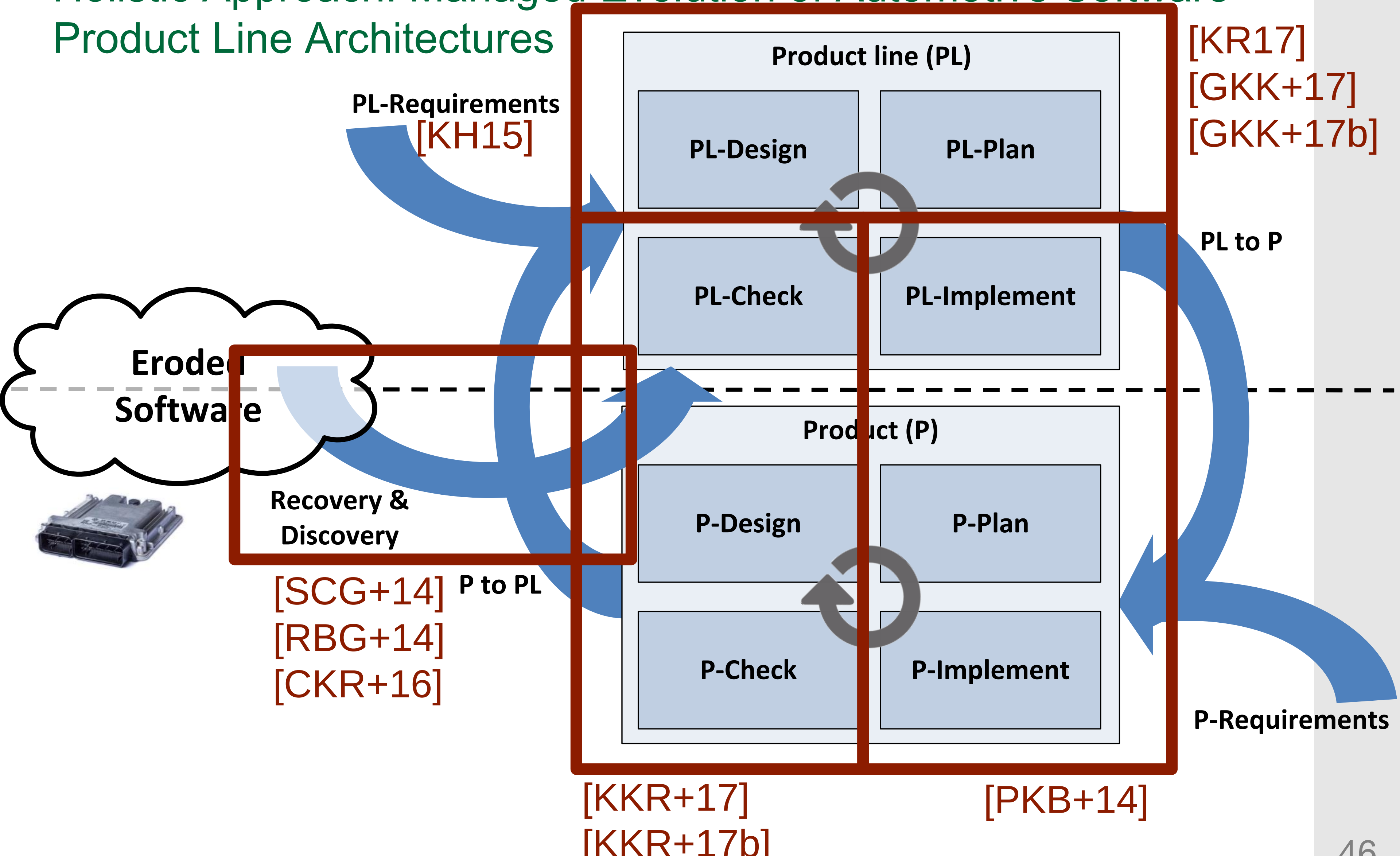
Digital Transformation - Transformation to a Digital Agile Organization





Reference: Dr. Katrin Allmendinger and Günther Thoma: Das Agile Unternehmen, see www.boeckler.de/pdf/v_2016_11_22_allmendinger_thoma.pdf

Holistic Approach: Managed Evolution of Automotive Software Product Line Architectures



References: List of Own Publications

[CKR+16]	B. Cool, C. Knieke, A. Rausch, M. Schindler, A. Strasser, M. Vogel, O. Brox, S. Jauns-Seyfried, "From Product Architectures to a Managed Automotive Software Product Line Architecture," in Proceedings of the 31st Annual ACM Symposium on Applied Computing, ser. SAC’16. ACM, 2016, pp. 1350–1353.
[GKK+17]	A. Grewe, C. Knieke, M. Körner, A. Rausch, M. Schindler, A. Strasser, M. Vogel, “Automotive Software Systems Evolution: Planning and Evolving Product Line Architectures,” in Special Track: Managed Adaptive Automotive Product Line Development (MAAPL), along with ADAPTIVE 2017. IARIA XPS Press, 2017, pp. 53–62.
[KH15]	C. Knieke, M. Huhn, "Semantic Foundation and Validation of Live Activity Diagrams," Nordic Journal of Computing, vol. 15, no. 2, 2015, pp. 112–140.
[KKR+17]	C. Knieke, M. Körner, A. Rausch, M. Schindler, A. Strasser, M. Vogel, “A Holistic Approach for Managed Evolution of Automotive Software Product Line Architectures,” in Special Track: Managed Adaptive Automotive Product Line Development (MAAPL), along with ADAPTIVE 2017. IARIA XPS Press, 2017, pp. 43–52.
[KR17]	C. Knieke, A. Rausch: "MAAPL: Managed Adaptive Automotive Product Line Development", Editorial of Special Track MAAPL along with ADAPTIVE 2017. IARIA XPS Press, 2017.
[KSGR12]	C. Knieke, B. Schindler, U. Goltz, A. Rausch, "Defining Domain Specific OperationalSemantics for Activity Diagrams," IfI Technical Report Series, IfI-12-04, Department of Informatics,TU Clausthal, 2012.
[PKB+14]	H. Peters, C. Knieke, O. Brox, S. Jauns-Seyfried, M. Krämer, A. Schulze, "A Test-driven Approach for Model-based Development of Powertrain Functions," in Agile Processes in Software Engineering and Extreme Programming. 15th International Conference on Agile Software Development, XP 2014. Springer-Verlag, 2014, pp. 294–301.
[RBG+14]	A. Rausch, O. Brox, A. Grewe, M. Ibe, S. Jauns-Seyfried, C. Knieke, M. Körner, S. Küpper, M. Mauritz, H. Peters, A. Strasser, M. Vogel, N. Weiss, "Managed and Continuous Evolution of Dependable Automotive Software Systems," in Proceedings of the 10th Symposium on Automotive Powertrain Control Systems, 2014, pp. 15–51.
[SCG+14]	A. Strasser, B. Cool, C. Gernert, C. Knieke, M. Körner, D. Niebuhr, H. Peters, A. Rausch, O. Brox, S. Jauns-Seyfried, H. Jelden, S. Klie, M. Krämer, "Mastering Erosion of Software Architectures in Automotive Software Product Lines," in Proceedings of the 40th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2014), ser. LNCS, vol. 8327. Springer, 2014, pp. 491–502.

Questions and Discussion