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
Short resume of the presenter

- 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

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


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

5. How to Cope with the Challenges of the Digital Transformation
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** runtime!)
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

```
int main(void) {
    #if a == 1
        //Block_1
    #else
        //Block_2
    #endif
    #if b == 1
        //Block_3
    #endif
    if (x == 1) {
        if (y == 1) {
            //Block_4
        } else {
            //Block_5
        }
    } else {
        //Block_6
    }
    return EXIT_SUCCESS;
}
```

```
int main(void) {
    #if a == 1
        //Block_1
    #else
        //Block_2
    #endif
    #if b == 1
        //Block_3
    #endif
    if (x == 1) {
        if (y == 1) {
            //Block_4
        } else {
            //Block_5
        }
    } else {
        //Block_6
    }
    return EXIT_SUCCESS;
}
```

```
int main(void) {
    #if a == 1
        //Block_1
    #else
        //Block_2
    #endif
    #if b == 1
        //Block_3
    #endif
    if (x == 1) {
        if (y == 1) {
            //Block_4
        } else {
            //Block_5
        }
    } else {
        //Block_6
    }
    return EXIT_SUCCESS;
}
```
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


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

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

- Domain Product Management
- Domain Realization

ISO 26262 – Bidirectional Traceability

Problem Space | Solution Space

Domain Engineering

- Domain Product Management
- Domain Realization

Appl. Engineering

- Appl. Product Management
- Application Req. Eng.
- Application Design
- Application Realization
- Application Testing

Software Product N
Challenge: Modelling of Requirements and Design Artifacts

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
Challenge: Variability and Configurability

- Domain Product Management
- Domain Req. Eng.
- Domain Design
- Domain Realization
- Domain Testing

- Domain Engineering
- Application Req. Eng.
- Application Design
- Application Realization
- Application Testing

Problem Space

Solution Space

Domain Artifacts

Software Product N
Challenge: Traceability of all Development Artifacts
Challenge: Lifecycle Management
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


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

5. How to Cope with the Challenges of the Digital Transformation
Importance of Modelling: Many Reasons for Introducing Model-based Engineering

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

- Uncordinated communication between components
- Mutual coordination
- Functions replicated in another context

➔ Growing of accidental complexity
➔ Huge effort of maintenance and further development

There is/was a plan
Importance of Architecture: Real World Example “Longitudinal Dynamics Torque Coordination”

- Some facts about the power train software system:

<table>
<thead>
<tr>
<th>Element Type*</th>
<th>Count (in 2015)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projects (versioned)</td>
<td>6.533</td>
</tr>
<tr>
<td>Modules (versioned)</td>
<td>21.734</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Element Type** (Project Example: MC6)</th>
<th>Count (in 2017)</th>
</tr>
</thead>
<tbody>
<tr>
<td>System constants (SC)</td>
<td>~18.000</td>
</tr>
<tr>
<td>Application parameters</td>
<td>~80.000</td>
</tr>
</tbody>
</table>

*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


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
- ...

Engine Type
Gear Type
Actuators
Sensors
Product Definition by Feature Trees

Feature Tree

Software Product Line Architecture

Product Definition: Missing Modelling Concept and Impact to Architecture Configuration
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


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

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

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


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

5. How to Cope with the Challenges of the Digital Transformation
Modification to Requirement 9

Requirements

Architecture
Modification to Component K11
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


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

5. How to Cope with the Challenges of the Digital Transformation
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
Handling Variability in the Implementation Artifacts
Approach: Parameterized Artifacts and Propagation of Variability Decisions including Derivation of Traces

Benefits: Enabling Traceability + Controlling Variability + Managing Evolution
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


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

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

<table>
<thead>
<tr>
<th></th>
<th>Count in 2012</th>
<th>Count in 2016</th>
<th>Number of versions</th>
<th>Average number of versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical architecture elements</td>
<td>10</td>
<td>15</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>Module architecture elements</td>
<td>10</td>
<td>15</td>
<td>58</td>
<td>4</td>
</tr>
<tr>
<td>Projects</td>
<td>1</td>
<td>21</td>
<td>146</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Number of versions used in projects</th>
<th>Cumulated number of versions used over all project versions</th>
<th>Average degree of reuse of each version</th>
<th>Number of used design configurations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module architecture elements</td>
<td>46</td>
<td>1611</td>
<td>35</td>
<td>n/a</td>
</tr>
<tr>
<td>Projects</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>14</td>
</tr>
</tbody>
</table>

➔ 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)
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


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

5. How to Cope with the Challenges of the Digital Transformation
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, …
The Heart of the Automobile in the 21. Century will be the Self-Driving-System

Engine as heart of the automobile

Self-Driving-System as heart of the automobile

Johann Jungwirth
Chief Digital Officer
Volkswagen Group
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

Product line (PL)
- PL-Design
- PL-Plan
- PL-Check
- PL-Implement

Product (P)
- P-Design
- P-Plan
- P-Check
- P-Implement

PL-Requirements
- [KH15]

Eroded Software
- [SCG+14]
- [RBG+14]
- [CKR+16]
- [KKR+17]
- [KKR+17b]
- [KR17]
- [GKK+17]
- [GKK+17b]
- [PKB+14]

Recovery & Discovery

PL to P

P to PL
## References: List of Own Publications

<table>
<thead>
<tr>
<th>Reference</th>
<th>Title and Details</th>
</tr>
</thead>
</table>
Questions and Discussion