Model-based Management of Software Platforms

Vector Congress, 20th of November 2018, Stuttgart
Agenda

- Project and Platform Driven Development
  - Reasons for Variants
  - Platform Process
  - Summary
Project and Platform Driven Development

Independent Software Projects

**Benefits:**
- Independent release planning
- Different SW suppliers possible
- Flexible SW architectures and functions
- Independent design and development processes

**Consequences:**
- Multiple development
- Multiple maintenance
- Multiple quality assurance of similar functions
Common Software Platform

Benefits
- High quality by continuous improvements and multiple usage
- High flexibility and agile creation of new variants and alternatives

Challenges
- Long term commitment to platform
- Tailoring of projects by variant management

Software Platform: One Software for Everything

Vehicle 1
Vehicle 2
Vehicle 3
Functional Variants

- Product Management Task
  - Definition of feature model
  - Definition of variants

- Task of Architects
  - Definition of functional architecture and interfaces
  - Definition of SW architecture and variant concepts
  - Mandatory guidelines for detailed SW design of functional components, e.g. exterior lighting

<table>
<thead>
<tr>
<th>Halogen</th>
<th>Xenon</th>
<th>LED</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vehicle 1</td>
<td>✔️</td>
<td>✖️</td>
</tr>
<tr>
<td>Vehicle 2</td>
<td>✖️</td>
<td>✔️</td>
</tr>
<tr>
<td>Vehicle 3</td>
<td>✖️</td>
<td>✖️</td>
</tr>
</tbody>
</table>

Variant 1: Vehicle 1
Variant 2: Vehicle 2
Variant 3: Vehicle 3
Variant 4: Vehicle 1, Vehicle 3
Variant 5: Vehicle 1, Vehicle 2, Vehicle 3
Technology Variants

 Reasons for Variants

Technology variants need abstraction concepts driven by:

> SW platforms (AUTOSAR Classic, Adaptive, ...)

> HW abstraction (drivers for LED cluster, ...)

<table>
<thead>
<tr>
<th>Vehicle 1</th>
<th>Vehicle 2</th>
<th>Vehicle 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTOSAR Classic</td>
<td>AUTOSAR Classic</td>
<td>AUTOSAR Adaptive</td>
</tr>
<tr>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✗</td>
<td>❌</td>
</tr>
</tbody>
</table>
Teamwork: SW Architecture, Design and Implementation

- Function Designer
- SW Architect
- SW Designer

Use Cases
- Use Case 1
- Use Case 2

Customer Features
- CF 1
- CF 2
- CF 3

Requirements
- Requirement 1
- Requirement 2
- Requirement 3

Logical Function Architecture
- Logical Domain A
- Logical Domain B

Service-Oriented Design
- Service 1
- Service 2

Structured Software System Design
- Application SWC Type 1
- Application SWC Type 2

Internal Behavior
- Processor
- Runnable 1
- Runnable 2

Implementation
- Package 1
- Package 2
- File A
- File B
- File C

Implementation in C:
variation by "#ifdef..."

Implementation in C++:
variant pattern, e.g.
inheritance
Platform Process

Platform Releases and Project Releases

Software Platform: One Software for Everything

**Platform Releases**

- **Project 1 Releases**
- **Project 2 Releases**
- **Project 3 Releases**

**Branch (Top-down approach)**

R1: Reuse
R2: Update
R3: Update
R4: Reuse

7/10
Platform Process

PREEvision Engineering Backbone

- Platform approach needs engineering backbone
  - Full traceability
  - Change and release management
  - Organization of teamwork

- Data management
  - Domain language
  - Reuse and branch
  - Product line management

- Integration of supplier SWCs
  - Important for central computing platforms

- PREEvision has core functionality to support Model-based Management of Software Platforms
Model-based Management of Software Platforms

6. PREEvision as Engineering & Collaboration Tool
For more information about Vector and our products please visit

www.vector.com

Author:
Dr.-Ing. Clemens Reichmann
Vector Germany