Function Development with PREEvision

PREEvision User Day 2019
Agenda

- Overview
  - Function Development Today
  - Function Development in the Future
  - Summary and Next Steps
Top Level Automotive Design Flow

Requirements Engineering and Management

- Functional Architecture Design
- SOA Design
- Software Design
- Physical Architecture Design
- Physical System Design
- COM Design
- WH Architecture Design
- WH Series Design

Data Management, Collaboration, Platform Management, Safety, Change Management

Overview
Top Level Automotive Design Flow

- Data Management, Collaboration, Platform Management, Safety, Change Management
- Requirements Engineering and Management
  - Functional Architecture Design
  - SOA Design
  - Software Design
  - Physical Architecture Design
  - Physical System Design
  - COM Design
  - WH Architecture Design
  - WH Series Design

- Test Engineering and Management

Legend:
- Tasks of Function Designer/ System Responsible
- Interactions
Overview

Function Development Flow in Detail

- Function Designer/System Responsible
- SW Architect
- SW Developer
- Component Supplier
- Test Engineer

Legend:
- Tasks of Function Designer/System Responsible
- Interactions

Component Specifications:
- REQ SPEC DOC
- REQIF
- ARXML
- TEST SPEC DOC
- ...

- Functional Architecture Design
- System SW Architecture Design
- Comp. SW Architecture Design
- Software Component Design
- SW Implementation
- Component and System Test

- Physical Architecture Design
- Physical System Design
- COM Design
- HW Implementation
- WH Architecture Design
- WH Series Design

- WH Architect
- Communication Designer
- Component Responsible
Tasks of the Function Designer:

- **Functional Architecture Design**
  - Logical function design from the sensors to the actuators
  - Behavior design of logical function components
  - Functional safety concept
  - Functional diagnostics concept

Tasks of the System Responsible:

- **Physical System Design**
  - Map function components to HW components
  - Define component software and component hardware needs from system perspective
  - Define communication needs
  - Define wiring harness needs
  - Define system variants
  - Provide system test specifications

- **Implementation Planning and Tracking**
  - Plan stepwise implementation in line with vehicle network and component releases
  - Provide staged system and component specifications

Overview

Functional Architecture Design and Physical System Design
Agenda

Overview

- Function Development Today
  
  Function Development in the Future
  
  Summary and Next Steps
**Physical Architectures Today: Design Space for Function Development**

**E/E Architecture:**
- Onboard

**SW Architecture:**
- Signal-oriented
- Service-oriented (only for Ethernet and Diagnostics)

**Software Platform:**
- AUTOSAR Classic

Design Workflow AUTOSAR Classic
- "service oriented"

Design Workflow AUTOSAR Classic
- "signal oriented"
The Function Designer considers and extends the **Feature Model** of the vehicle family under development:
- Vehicle features - can be experienced by the vehicle user
- Safety features
- Technical features

**Feature Dependencies**
- Most relevant are:
  - Exclusive OR (Xor)
  - Needs (Req)
  - Optional

**Feature Chains**
- Needs/needed by
Logical Architecture Design and Mapping of Functions to Components

AUTOSAR Classic Ports are used:
- Sender/Receiver Ports and interfaces
- Client/Server Ports and interfaces

Example:
- Distributed function
  - implemented with a CAN network
  - specified with Sender/Receiver Ports.

The Function Designer defines ...
- ... the logical functions from the sensors to the actuators
- ... the functional safety and functional diagnostics concept
- ... the mapping of the functions to the E/E components.
Logical Architecture Design and Mapping of Functions to Components

The result is the System/Component Matrix:

<table>
<thead>
<tr>
<th>Component</th>
<th>System X</th>
<th>System Y</th>
<th>System Z</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECU A</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Sensor A11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor A12</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>ECU A1</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Actuator A11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor A21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ECU A2</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Actuator A21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Component</th>
<th>System X</th>
<th>System Y</th>
<th>System Z</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECU A</td>
<td></td>
<td>Function Y1</td>
<td>Function Z3</td>
<td></td>
</tr>
<tr>
<td>Sensor A11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor A12</td>
<td></td>
<td></td>
<td>Sense Z1</td>
<td></td>
</tr>
<tr>
<td>ECU A1</td>
<td></td>
<td>Function X3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Actuator A11</td>
<td></td>
<td>Actuation X4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor A21</td>
<td></td>
<td>Sense X1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ECU A2</td>
<td></td>
<td>Function X2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Actuator A21</td>
<td></td>
<td>Actuation X5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note:
- Unclear Responsibilities become obvious!
Logical Architecture Design

Function Development Today

Dedicated Modeling Concepts in the Logical Architecture:

- Different block types
  - Sense, Logical Function, Actuation
  - Building Block, Logical Domain
  - Environment Function, ...
- Type/Instance concept
- Activity Chain
  - they can overlaj
  - often used to map Functionality to Customer Features
- Logical Function Package
- Ports and Assembly Connections
  - Sender/Receiver Ports, Client/Server Ports
  - Environment Ports, ...
Behavior of Logical Functions

- The Function Designer defines also the behavior of the Logical Functions.
- This is supported with UML State Diagrams.
- Data Elements of Sender Receiver interfaces assigned to Ports of the Logical Function can be used in the Transition Conditions of the State Machine.

Also the Behavior of Software Components can be specified with State Diagrams.

Further UML Diagrams for behavior specification are on the PREEvision Roadmap!
Function Development Today

Wiring Needs for Physical Systems

- The System Responsible has to define the needs for the Wiring Harness Design
  - especially the wiring needs for sensors and actuators
  - possible with the Schematics Layer
- In the Schematics Diagram the needed pins of the components and the schematic connections can be specified.
- Of course, the pins of the components need to be coordinated with the Component Responsible.
- Also variants have to be considered.
The Component Responsible defines:
- ... the internal component structure
- ... the headers and the pins of the component

Component Diagram:

<table>
<thead>
<tr>
<th>Index</th>
<th>Component Pin</th>
<th>Pin Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Header Pin823</td>
<td>17</td>
</tr>
<tr>
<td>1</td>
<td>Header Pin821</td>
<td>19</td>
</tr>
<tr>
<td>2</td>
<td>Header Pin822</td>
<td>35</td>
</tr>
<tr>
<td>3</td>
<td>Header Pin824</td>
<td>36</td>
</tr>
</tbody>
</table>
Variants of Logical and Physical Systems

- Design Space is always a Product Line
- Widely accepted Variant Management Approach:
  - Take care only for the differences of the variants, …
  - … not for the commonalities (default)
- By defining Variation Points with
  - Variation Point Sets and
  - Variant Conditions
- based on System Constants and Literals

Wide accepted Variant Management Approach:
- Take care only for the differences of the variants, …
- … not for the commonalities (default)
- By defining Variation Points with
  - Variation Point Sets and
  - Variant Conditions
- based on System Constants and Literals

Compact Class
- Common
  - Compact Class only
  - Middle Class only

Middle Class

Function Development Today

Vector Functional Architecture Design
Design Workflows (1/2)

Alternative 1

- Abstract functional design: No direct contribution to the vehicle
- Concrete SW design: Often done at suppliers, often only top compositions defined by OEM
- SW to HW mapping: Clearly the OEMs responsibility
- Communication design: Clearly the OEMs responsibility

- Abstract logical function design:
  - Independent of implementation and highly reusable!
  - ... but often we see low motivation to maintain.
Alternative 2

- Concrete functional design ➔ Direct contribution to the vehicle, high motivation to maintain
- Function to HW mapping ➔ Clearly the OEMs responsibility
- Communication design ➔ Clearly the OEMs responsibility
- High level SW design ➔ Mostly done for the complete vehicle by the OEM (top compositions per ECU)
- Detailed SW design ➔ Optional: for inhouse development needed, otherwise at the supplier

- Concrete logical function design – including deployment aspects – drives communication
- Communication design and SW component design in parallel
- The more agile approach:
  - Detailed SW design can be changed as long as network communication is not affected!
Agenda

Overview
Function Development Today

- Function Development in the Future
Summary and Next Steps
Function Development in the Future

Physical Architectures in the Future: A bigger Design Space (1/2)

**E/E Architecture:**
- Onboard
- Offboard
- High Performance Computers

**SW Architecture:**
- Signal-oriented
- Service-oriented

**Software Platform:**
- AUTOSAR Adaptive
- AUTOSAR Classic
- Other Platforms

---

**AUTOSAR Adaptive**

**AUTOSAR Classic**
Physical Architectures in the Future: A bigger Design Space (2/2)

Many new options for Function Development

- Use and integrate Services in the Backend
- Run Algorithms with high performance needs on the High Performance Computers
- Update Functions over the air on AUTOSAR Adaptive ECUs
- Run safety relevant Functions on AUTOSAR Classic ECUs
- Provide remote Diagnostics Functions
- ...

Challenge

- How to cope with the Complexity!
For Service-oriented Architectures AUTOSAR Adaptive Ports are available in PREEvision 9.0:

- Service Ports and Service interfaces

**Example:**
- Distributed functions realized via an Ethernet network are specified with Service Ports.
The Service-oriented Architecture Design – as separate design layer – considers additional design aspects:

- Different possible scenarios after service discovery
- Alternative providers
- Several consumers

- All details of the necessary Service Interfaces have to be defined.
- Service dependencies have to be defined.

These are the tasks of a new role in the function development process:
- Service Architect
### Functional Architecture Design

- **Instance perspective (or object view)** for functions and systems including mappings of the functions to the HW components
- **Type view** for functions available, but in most cases of lower importance
  - Some exceptions: Window left/right, Mirror left/right, Seat left/right, ....
- **End-to-end view** of functional dependencies ... from the sensors to the actuators
- Communication needs for service oriented and signal oriented architectures

### Service Architecture Design

- **Type perspective (or class view)** for services and software components
- Alternative service providers and other service consumers are considered
- Service layers and Service Dependencies are defined
- The SOA layer is an abstract layer connecting the AUTOSAR Classic and AUTOSAR Adaptive sub systems in the vehicle ...
- ... and can serve as single source for the derivation of AUTOSAR Classic and AUTOSAR Adaptive SWC types.

### Mappings between Logical Architecture and Service Architecture

- Function → M – Participant
- Function Type – M – Service Provider/Consumer
Function Development in the Future

Component Software Architecture Design

- With AUTOSAR Adaptive the implementation changes from C to C++.
- High Performance Computers will have a more complex SW Architecture than ECUs so far.
- With UML Class Diagrams the internal structure of AUTOSAR Adaptive SWCs can be specified:
How will the Design Workflow look in Future?

- Currently discussions with many customers and users.
- Also concept discussions in AUTOSAR have started: „VFB++“, ...

- **Possible solutions:**
  - Logical function design – SOA design – SW design

- **We have 2 possible approaches in mind:**
  - Pragmatic approach
  - „New Thinking“
Pragmatic Approach

- If functional domains are using one SW Platform and are clearly separated, then...
- ... use AUTOSAR Classic Ports in the AUTOSAR Classic Signal-oriented areas
- ... use AUTOSAR Adaptive Ports in the AUTOSAR Classic and AUTOSAR Adaptive Service-oriented areas
Pragmatic Approach

- If functional domains
  - use different SW Platforms or
  - are not separated,
    then ...
- ... the Pragmatic Approach does not work!
Service Interface and Technology Mapping to AUTOSAR Classic

AUTOSAR Adaptive

- **Service Interface**
  - Method
  - Fire & Forget Method
  - Property
  - Event

AUTOSAR Classic

- Application SW Component (Service Provider)
  - Client/Server Interface
  - Sender/Receiver Interface
  - Client/Server Interface with GET_ and SET_ operation
  - Sender/Receiver Interface for change notification

1: Fire & Forget Method = Method without Return
2: Property = Field = Attribute

Abstraction

- Service Provider/Consumer Port
- Service Interface
- Event
- Sender/Receiver Port
- Sender/Receiver Interface
- Data Element
- Signal
The logical design uses only Service Ports

... and can be deployed to AUTOSAR Classic and AUTOSAR Adaptive ECUs.
Function Development in the Future

New Thinking

- The Logical Function Design is specified for the complete vehicle.
- The Service Map or Service Landscape is defined for the complete vehicle, too, with
  - Layers, clusters and dependencies
- The Software Architectures for AUTOSAR Classic and AUTOSAR Adaptive are derived from these Layers.
Agenda

Overview
Function Development Today
Function Development in the Future

> Summary and Next Steps
Summary and Next Steps

Function Development

The model-based design of
- Logical functions and
- Physical systems

enables the
Consistent derivation of
- Software needs
- Communication needs
- Component needs
- Wiring harness needs and
- System variants.

Additional specification done with
- Integrated requirements and
- Test specifications

Integrated change management
- Implementation and release planning in integration steps
- Ticket and change management
- Review and voting support
Summary and Next Steps

Function Development

- Today Function Development has a clear Onboard Design Space.
- In the near future a bigger Design Space will be available – with many new options!
- Challenge:
  - Master the complexity
  - Concepts are in discussion

<table>
<thead>
<tr>
<th>Logical Function Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service oriented Architecture</td>
</tr>
<tr>
<td>AUTOSAR Classic SW Architecture</td>
</tr>
<tr>
<td>AUTOSAR Adaptive SW Architecture</td>
</tr>
<tr>
<td>SW Component Behavior</td>
</tr>
<tr>
<td>SW Class Architecture &amp; SW Component Behavior</td>
</tr>
<tr>
<td>HW Network Architecture</td>
</tr>
<tr>
<td>Communication Designer</td>
</tr>
<tr>
<td>SW Developer</td>
</tr>
<tr>
<td>SW Architect</td>
</tr>
<tr>
<td>Service Architect</td>
</tr>
<tr>
<td>E/E Architect</td>
</tr>
<tr>
<td>Test Engineer</td>
</tr>
<tr>
<td>E/E System Responsible, Function Designer</td>
</tr>
<tr>
<td>Requirements</td>
</tr>
<tr>
<td>Logical Function Architecture</td>
</tr>
<tr>
<td>Software/Service Architecture</td>
</tr>
<tr>
<td>Hardware Architecture</td>
</tr>
<tr>
<td>Wiring Harness</td>
</tr>
</tbody>
</table>
For more information about Vector and our products please visit

www.vector.com

Author:
Jörg Schäuffele
Vector Germany