AUTOSAR标准与体系

主讲：丁旭阳 dxy@uestc.edu.cn

电子科技大学 嵌入式软件工程中心

2014/2/17

主要内容

1. 关于AUTOSAR
2. AUTOSAR标准文档
3. AUTOSAR方案与基本概念
4. AUTOSAR软件架构
5. AUTOSAR方法论

1. 关于AUTOSAR？

1.1 为什么需要AUTOSAR？

1.2 AUTOSAR组织

1.3 AUTOSAR定义

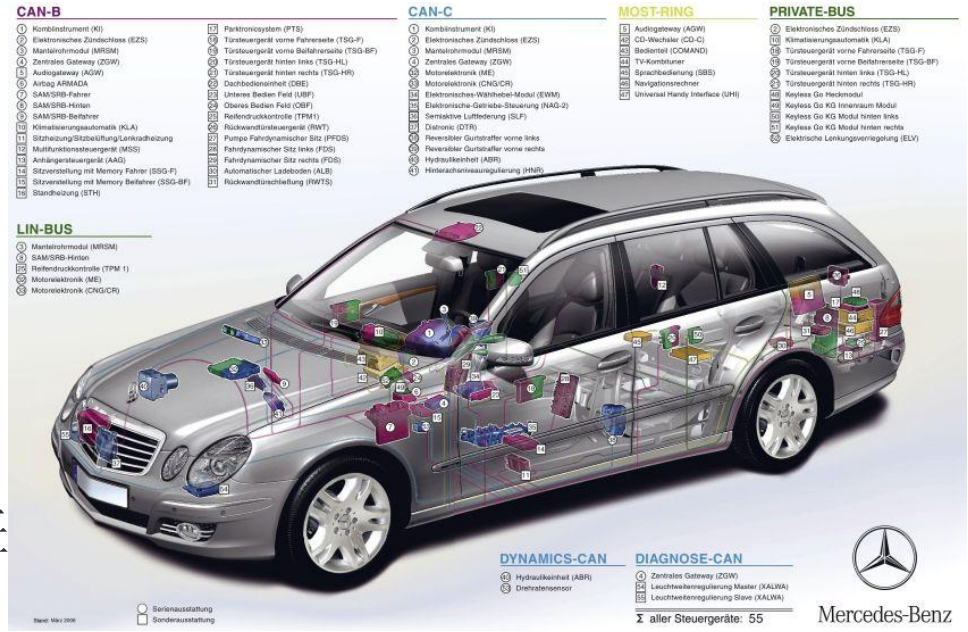
1.4 AUTOSAR特征与技术范围

1.5 AUTOSAR实现情况

1.1 为什么需要AUTOSAR?

- **复杂度**
  - ✓ 上百的功能函数
  - ✓ 50+ ECUs
  - ✓ 网络化的控制
  - ✓ 许多供应商
  - ✓ 异构的平台
- **软件集成面临的挑战**
  - ✓ 风险：可靠性，质量，责任
  - ✓ 开发和产品成本

55 ECUs & 7 Buses of 4 types with Gateways



source: Daimler-Chrysler

1.1 为什么需要AUTOSAR?

- 动机
  - ✓ 管理功能日益复杂的汽车E/E系统
  - ✓ 便于产品修改、更新和升级
  - ✓ 产品线内和跨产品线的方案可裁剪性
  - ✓ 提高E/E系统的质量和可靠性

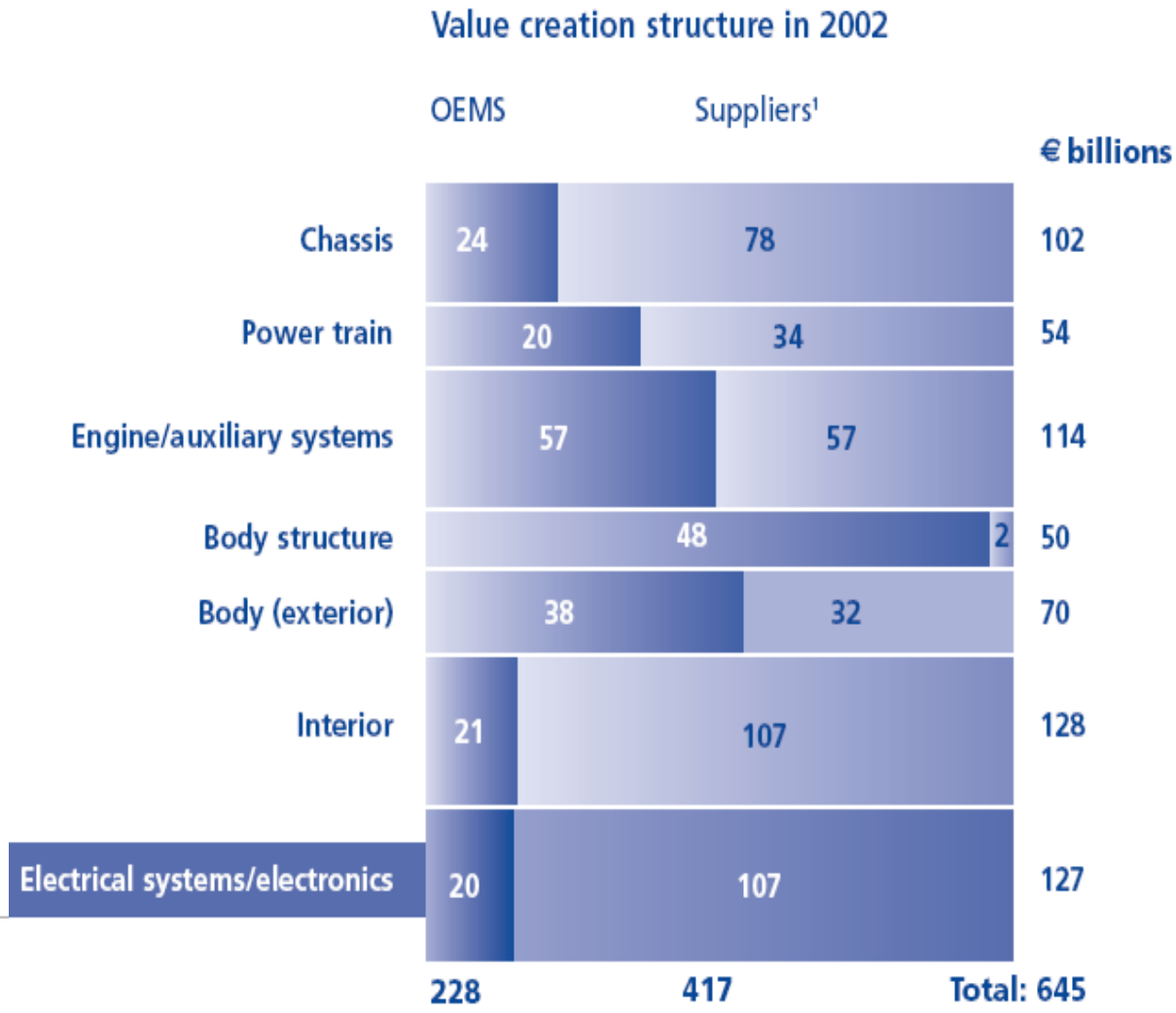
1.1 为什么需要AUTOSAR?

- 目标
  - ✓ 实现未来车载系统需求：可用性和安全性、软件升级/更新和可维护性
  - ✓ 增加功能集成和转移的可裁剪性和灵活性
  - ✓ 产品线上大量采用商业化的软硬件构件
  - ✓ 提高产品处理复杂功能和风险的能力
  - ✓ 可裁剪系统的成本优化
  - ✓ …

- AUTOSAR_RS_ProjectObjectives.pdf

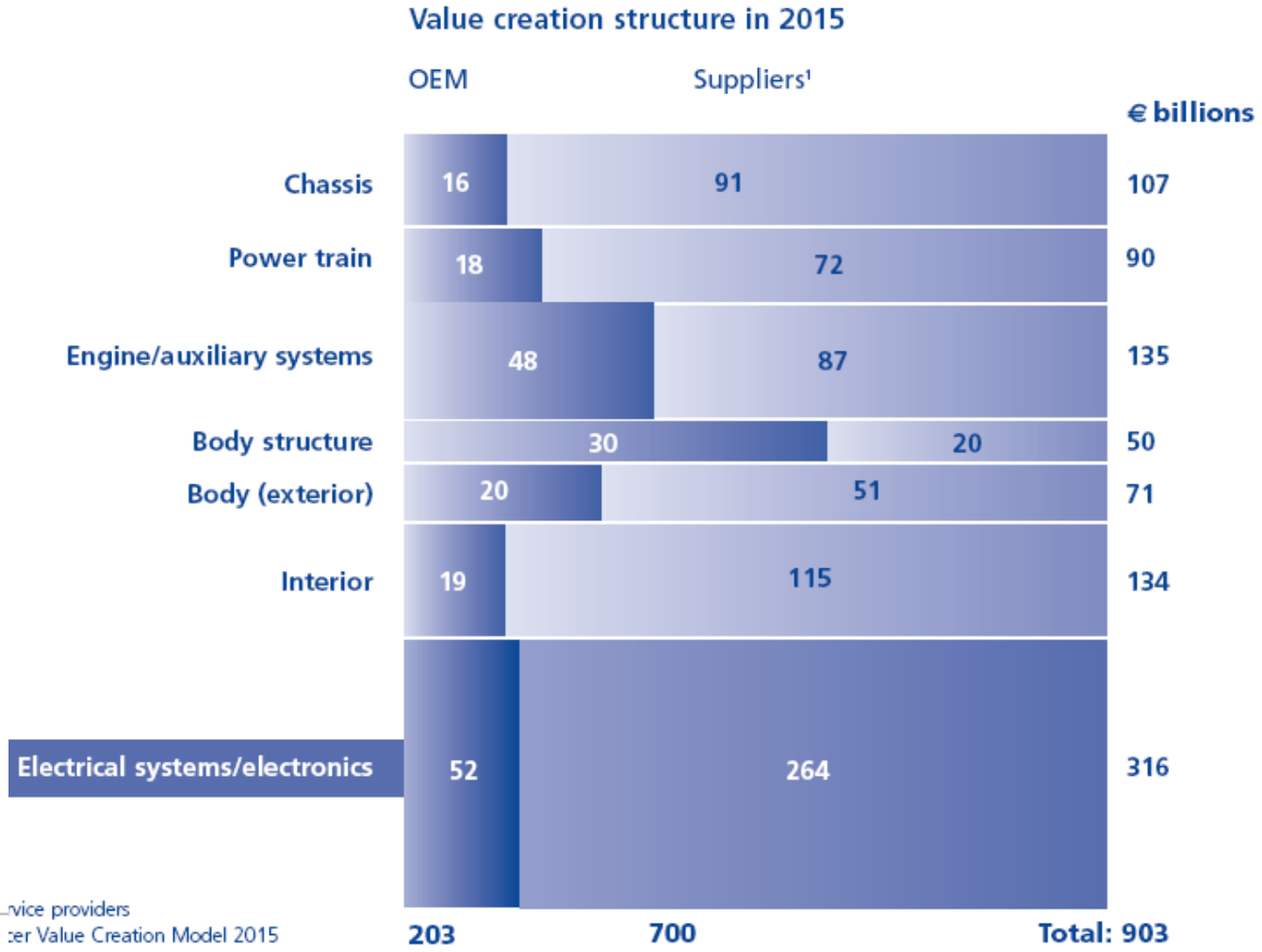
1.1 为什么需要AUTOSAR?

- “the coming age of collaboration in the automotive industry”



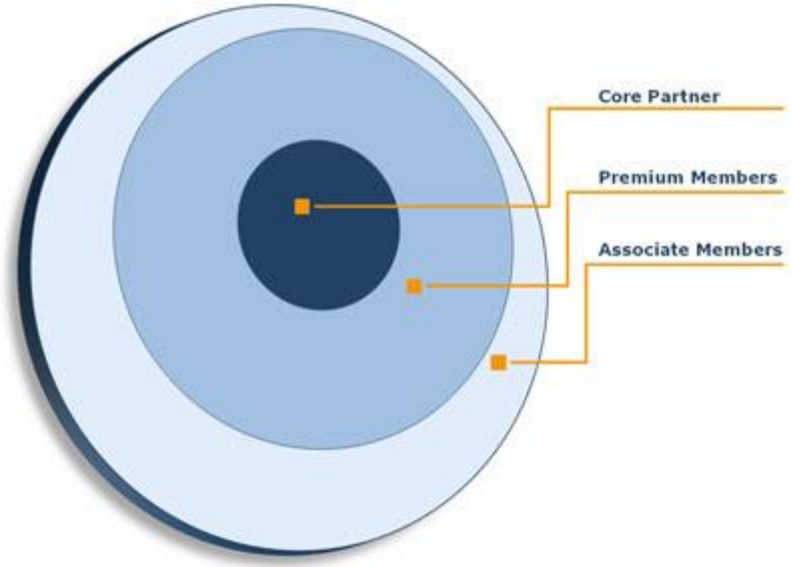
1.1 为什么需要AUTOSAR?

- “the coming age of collaboration in the automotive industry”



1.2 AUTOSAR组织

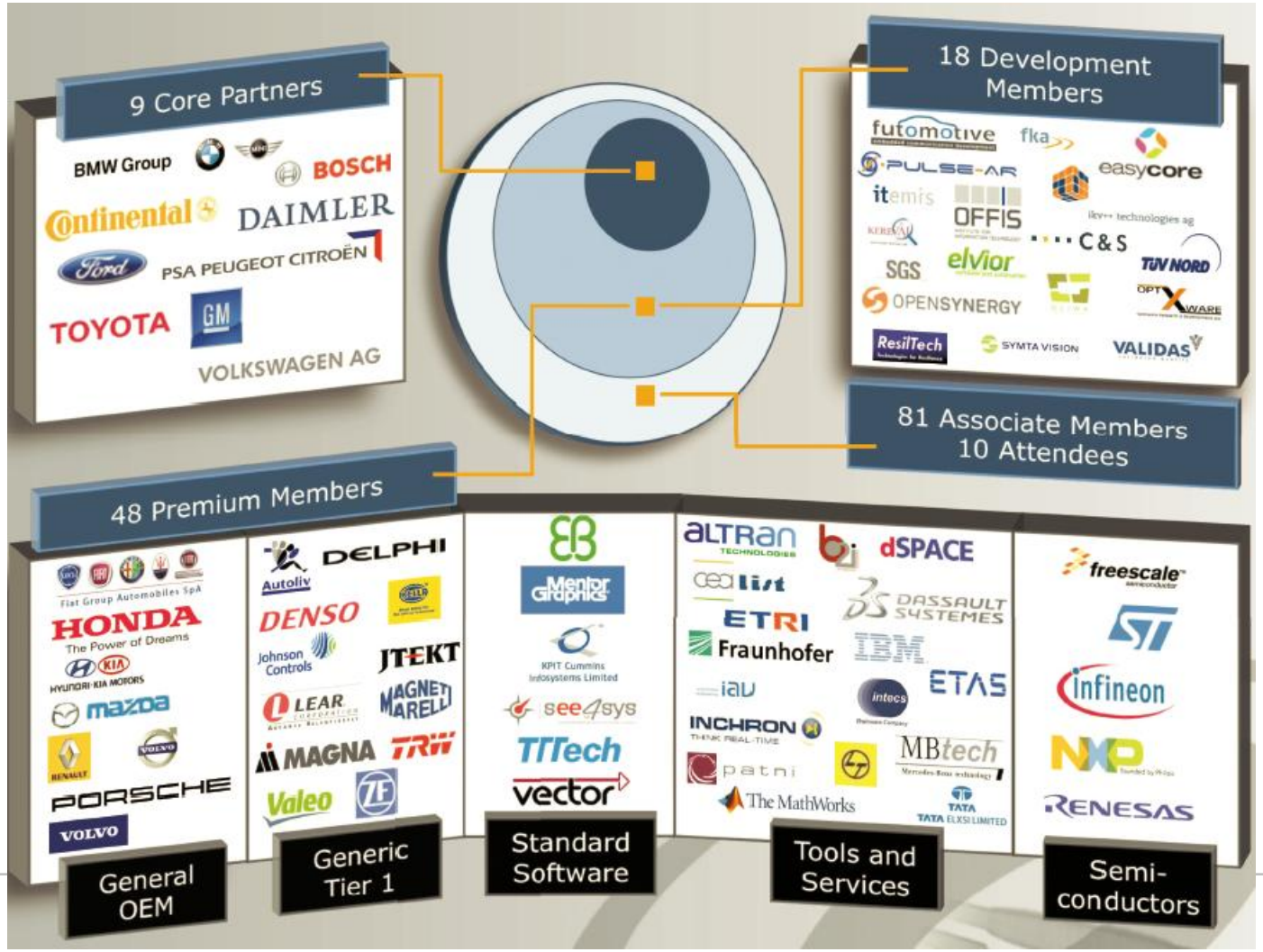
- 成立于2003年
- AUTOSAR 组织成员类型
  - 9 Core Partners (BMW, Bosch, Continental, Daimler, Ford, Opel, PSA Peugeot Citroën, Toyota, Volkswagen)
  - Premium Members
  - Associate Members
  - Development Members
  - Attendees



Source: www.autosar.org

1.2 AUTOSAR组织

Status: December 2011



1.3 AUTOSAR定义

- AUTOSAR = AUTomotive Open System Architecture
  - ✓ 汽车开放系统架构
  - ✓ 开放的、标准化的汽车软件架构
  - ✓ 由主流汽车OEMs,供应商以及工具开发商制定的开放标准
  - ✓ 提供基础设施帮助开发车载软件、用户界面和管理所有的应用领域
    - 基础系统功能的标准化
    - 适应不同车载平台
    - 可集成多个供应商的产品

1.4 AUTOSAR特征与技术范围

- 模块化与可配置性
  - ✓ 模块化的软件体系结构
  - ✓ 区分硬件相关与硬件无关的软件模块
  - ✓ 集成不同供应商提供的软件模块，增加功能复用
  - ✓ 软件模块的可转移性
  - ✓ 基于功能部署的情况，对每个ECU的软件基础平台进行资源优化的配置
  - ✓ 跨整个车载产品线的E/E系统的可伸缩性

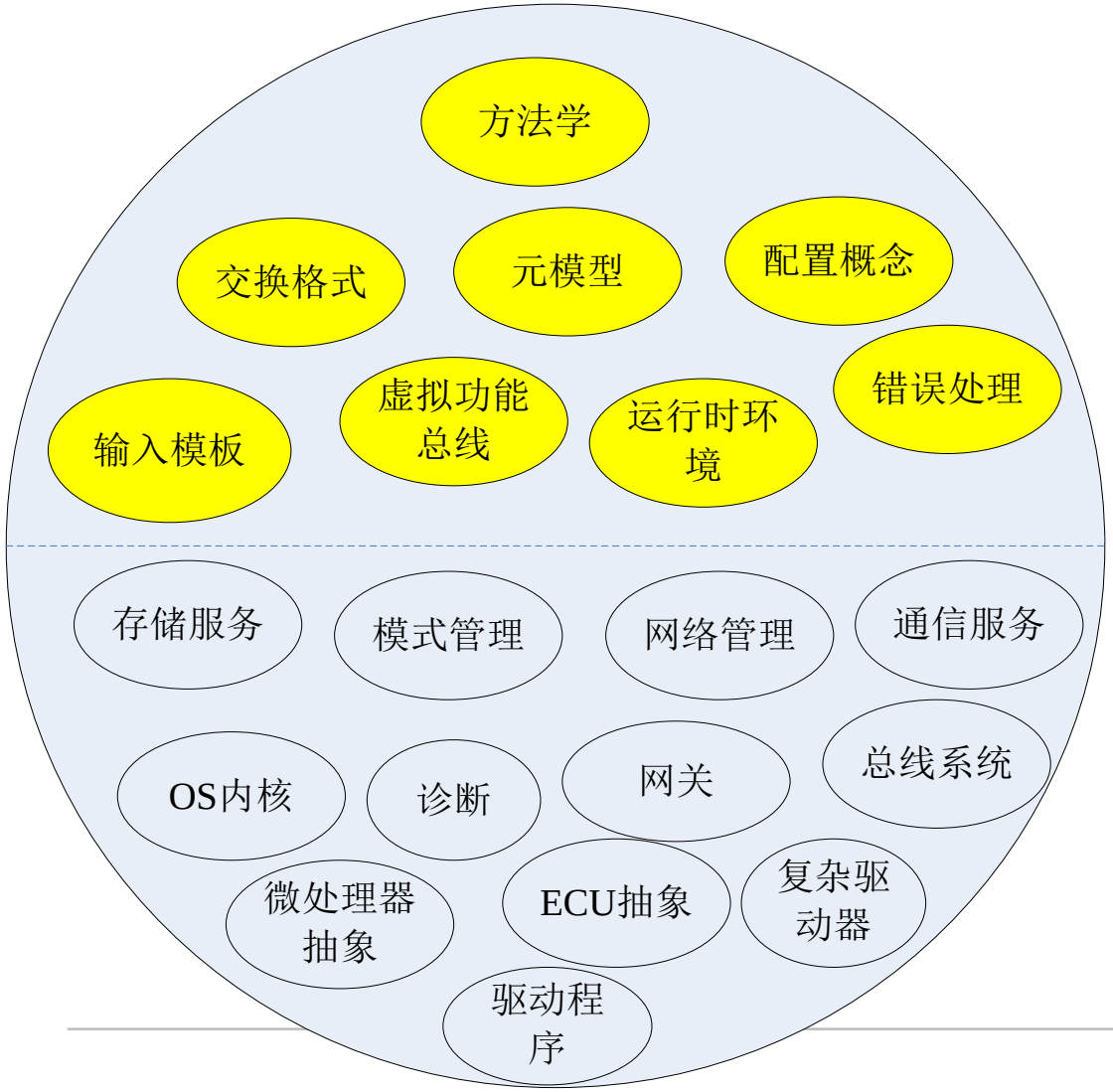
1.4 AUTOSAR特征与技术范围

- 标准化的接口
  - ✓ 标准化不同的API，分离AUTOSAR软件层
  - ✓ 便于功能化软件构件的封装
  - ✓ 定义软件构件的数据类型
  - ✓ 基础软件模块接口的标准化

1.4 AUTOSAR特征与技术范围

- 运行时环境(RTE)
  - ✓ 提供跨整个车载网络节点的ECU内和ECU间通信
  - ✓ 位于功能性的软件构件与基础软件模块间
  - ✓ 与ATUOSAR RTE连接的所有实体必须遵从AUTOSAR规范
  - ✓ 易集成定制的特定功能性软件模块

1.4 AUTOSAR特征与技术范围



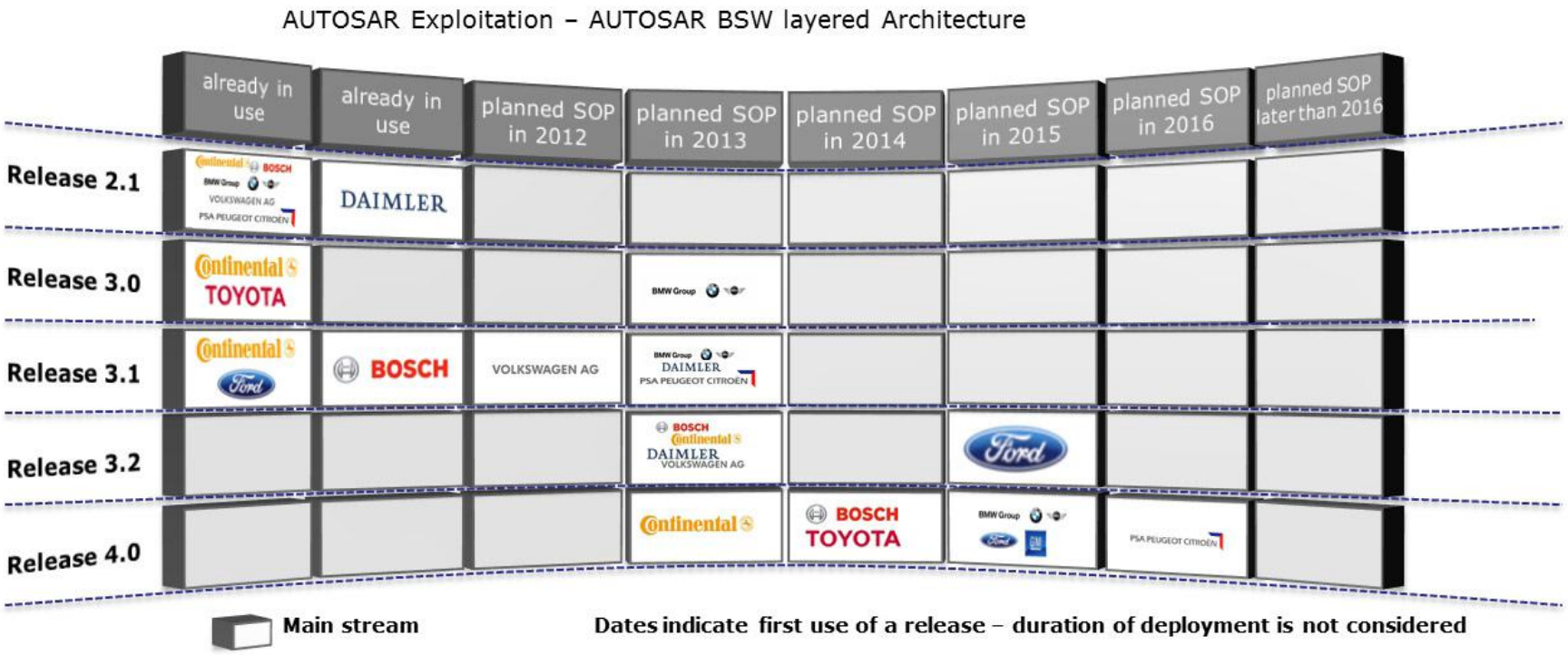
新概念

工业界现有的基础软件设计

1.5 AUTOSAR 实现情况

- 许多OEMs 和供应商已在大范围的应用中引入AUTOSAR
- 大多数核心成员将于2015年完成基础软件的迁移,使得基础软件完全遵从AUTOSAR规范

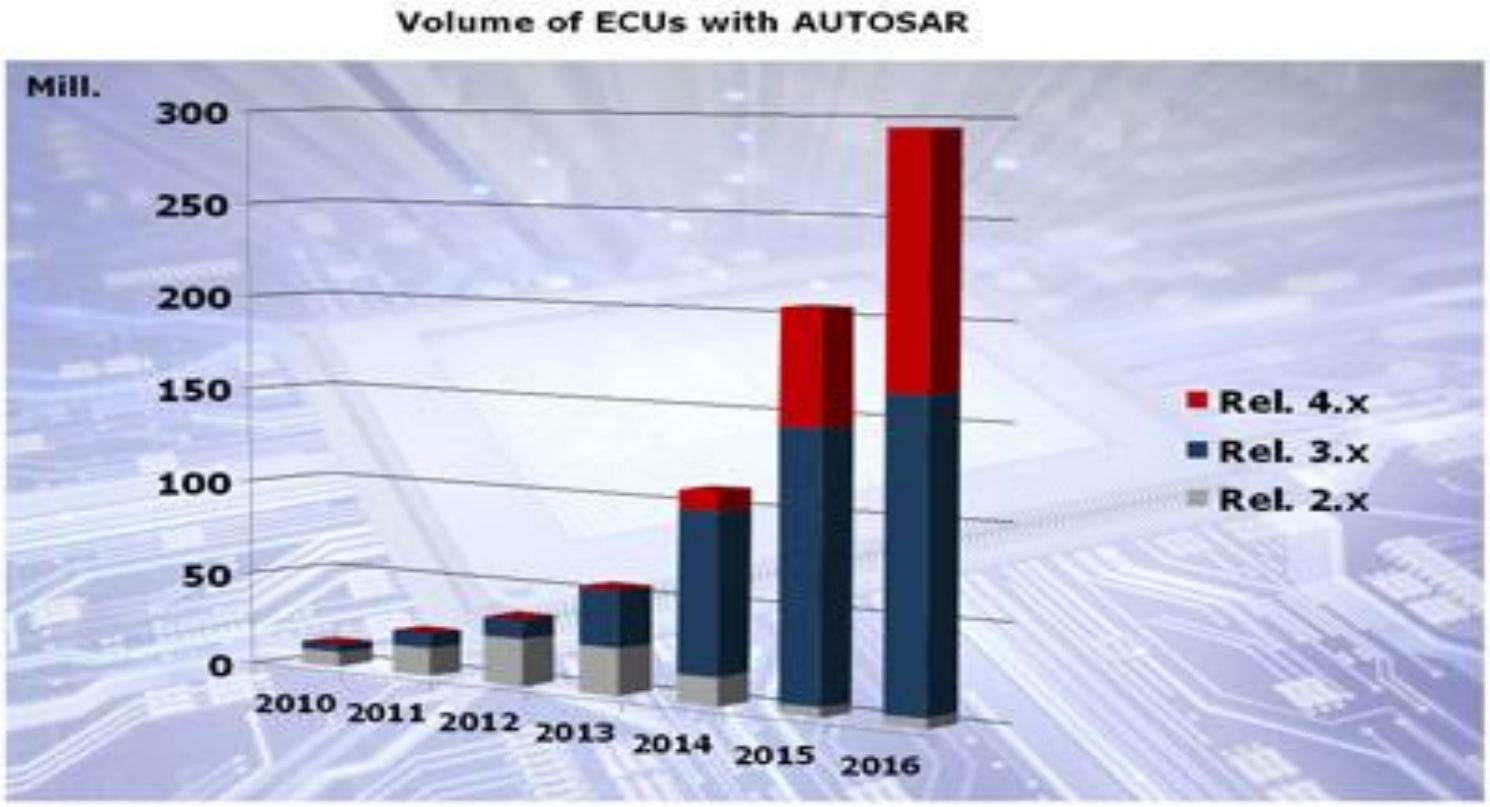
1.5 AUTOSAR 实现情况



The main stream of AUTOSAR releases are R3.x and R4.0

1.5 AUTOSAR 实现情况

- AUTOSAR核心成员在2011年生产了基于AUTOSAR架构的2千5百万颗ECU,并计划2016年生产大约3亿颗。



主要内容

1. 关于AUTOSAR
2. AUTOSAR标准文档
3. AUTOSAR方案与基本概念
4. AUTOSAR软件架构
5. AUTOSAR方法论

2. AUTOSAR标准文档

2.1 版本历史

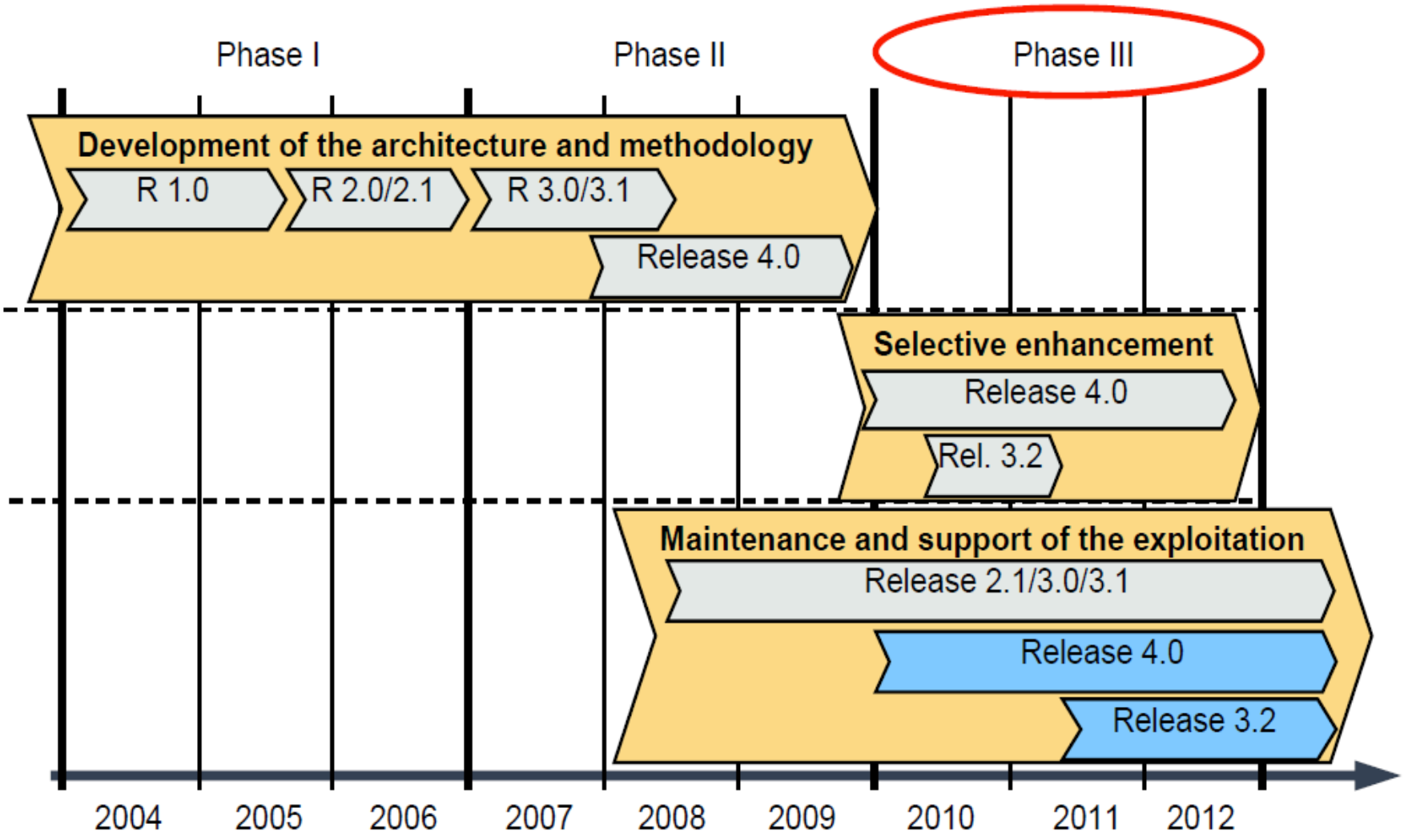
2.2 组织结构

2.3 文档类型

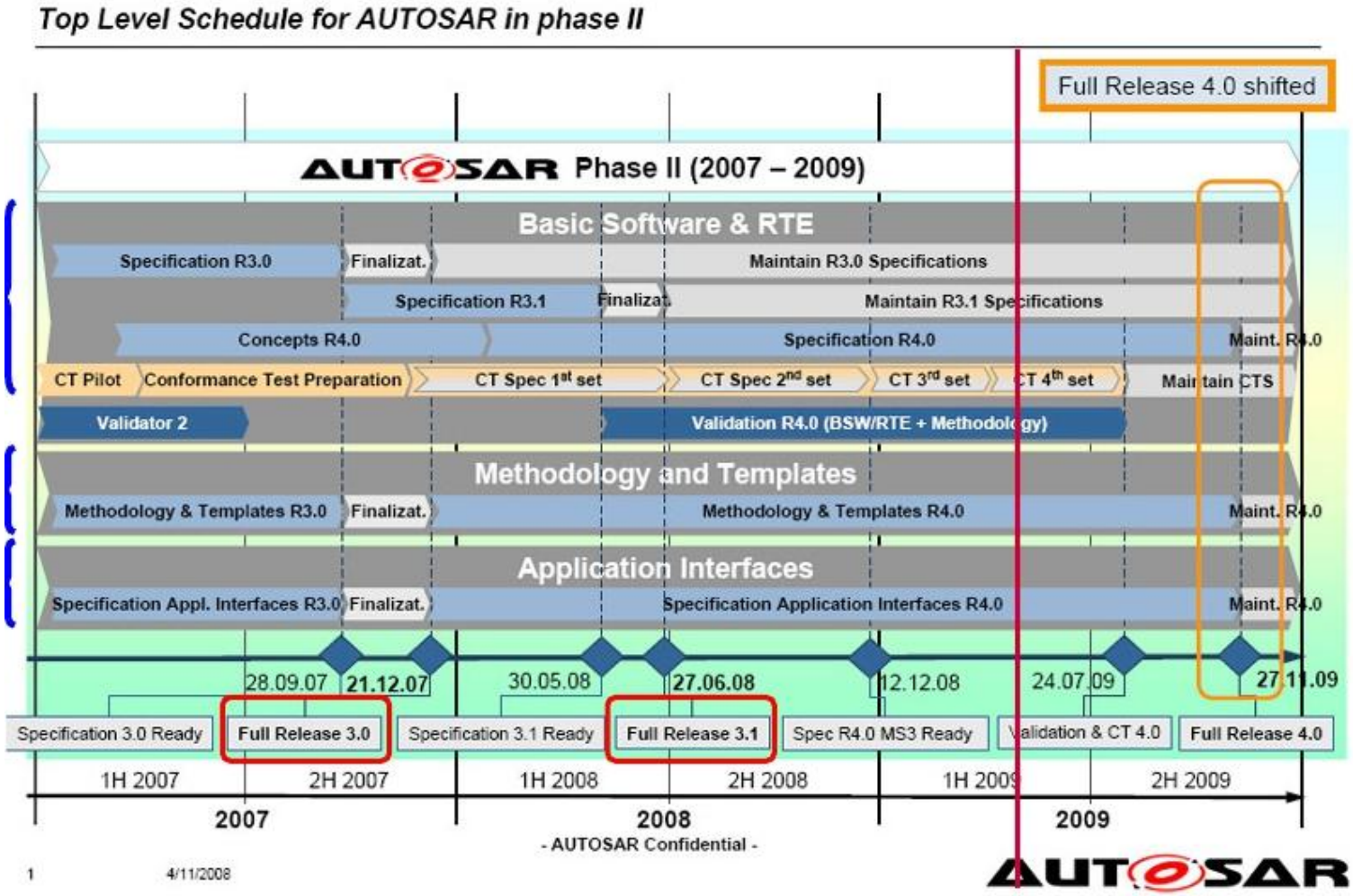
2.1 版本历史

- Release 2.0/2.1, Release 3.0, Release 3.1, Release 3.2, Release 4.0, …
- 3 Phases
  - ✓ Phase I (2004-2006)
  - ✓ Phase II (2007-2009)
  - ✓ Phase III (2010-2012)

2.1 版本历史


Phase I
Phase II
Phase III
Development of the architecture and methodology
R 1.0
R 2.0/2.1
R 3.0/3.1
Release 4.0
Selective enhancement
Release 4.0
Rel. 3.2
Maintenance and support of the exploitation
Release 2.1/3.0/3.1
Release 4.0
Release 3.2
2004
2005
2006
2007
2008
2009
2010
2011
2012

2.1 版本历史

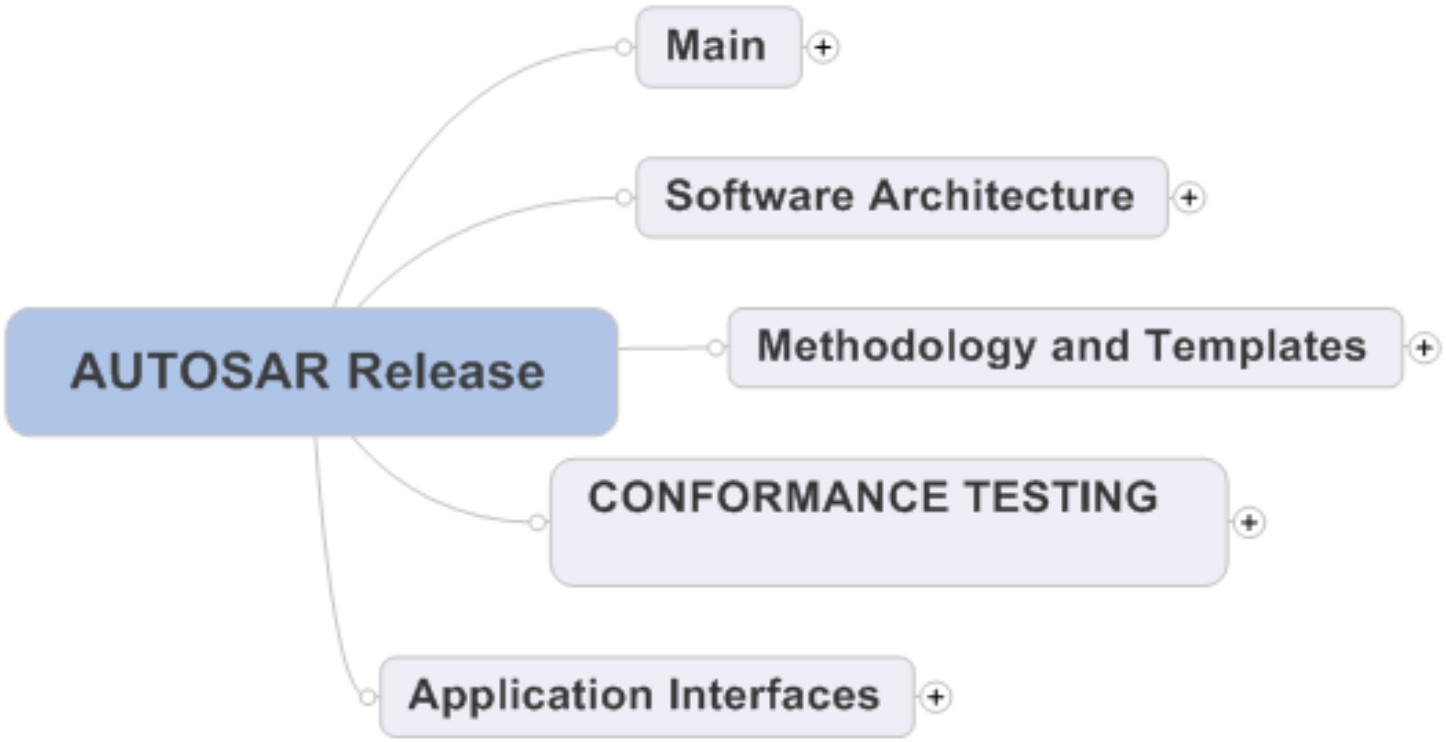


Specification and Test

Methodology and Templates

Application Interface

2.2 组织结构


AUTOSAR Release
Main
Software Architecture
Methodology and Templates
CONFORMANCE TESTING
Application Interfaces

2.3 文档类型

- AUTOSAR_TR_PredefinedNames.pdf

Short Name	Long Name	Document Type
CONC	Concept Document	Concept describing planned changes for the next minor or major release
CTCF	Configuration Settings	Configuration settings for the execution of conformance Tests
CTSP	Conformance Test Specification	Test specification and scripts for the execution of conformance tests
EXP	Explanation	Explanatory material discussing contents already shown in other documents

2.3 文档类型

Short Name	Long Name	Document Type
MMOD	MetaModel	Modeled contents (a model or generated from a model) on meta level 2 (Meta-Model)
MOD	Model	Modeled contents (a model or generated from a model) on meta level 1 (Model)
PD	Process Description	Description of process applied within AUTOSAR standardization activities
RS	Requirement Specification	Specification of requirements other than for software specifications

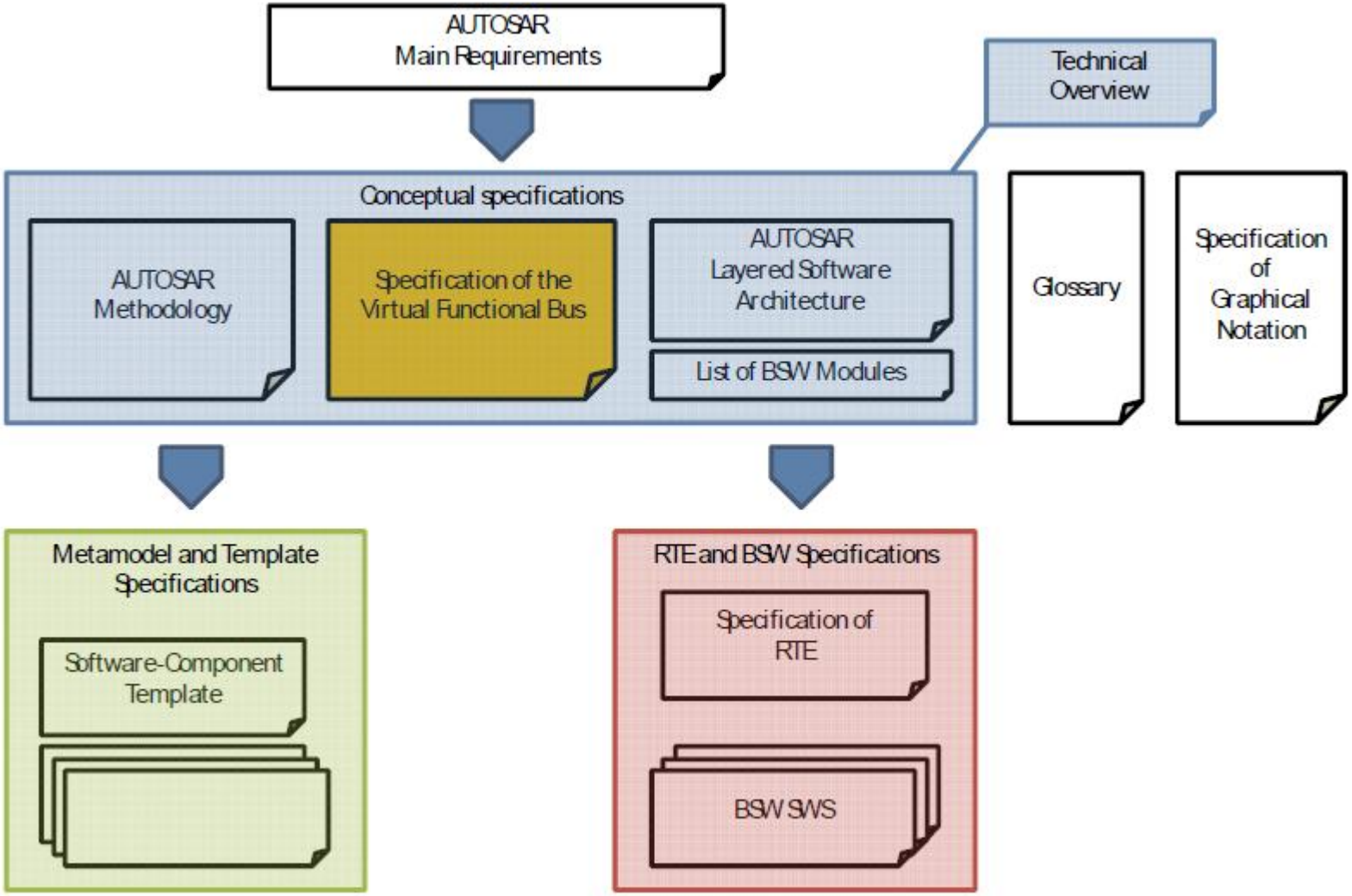
2.3 文档类型

Short Name	Long Name	Document Type
SRS	Software Requirement Specification	Specification of requirements for software specifications
SWS	Software Specification	Specification of AUTOSAR Software
TPS	Template Specification	Specification of AUTOSAR Templates, containing Meta model information, constraints etc.
TR	Technical Report	A general technical report describing arbitrary AUTOSAR related topics
UC	Use Case Specification	Specification of use cases from which requirements are derived.

2.3 文档类型

Short Name	Long Name	Document Type
ZAUX	Auxiliary material	Auxiliary files used internally for the creation of the standard. May be merged with ZSUPP.
ZGEN	Generated intermediate material	Generated intermediate products which are maintained in the SCM system of AUTOSAR and used internally for the creation of the standard
ZSUPP	Supplemental material	Supplementary material used internally for the creation of the standard

2.3 文档类型


AUTOSAR
Main Requirements
Technical
Overview
Conceptual specifications
AUTOSAR
Methodology
Specification of the
Virtual Functional Bus
AUTOSAR
Layered Software
Architecture
List of BSW Modules
Glossary
Specification
of
Graphical
Notation
Metamodel and Template
Specifications
Software-Component
Template
RTE and BSW Specifications
Specification of
RTE
BSW SWS

主要内容

1. 关于AUTOSAR
2. AUTOSAR标准文档
3. AUTOSAR方案与基本概念
4. AUTOSAR软件架构
5. AUTOSAR方法论

3. AUTOSAR方案与基本概念

3.1 方案

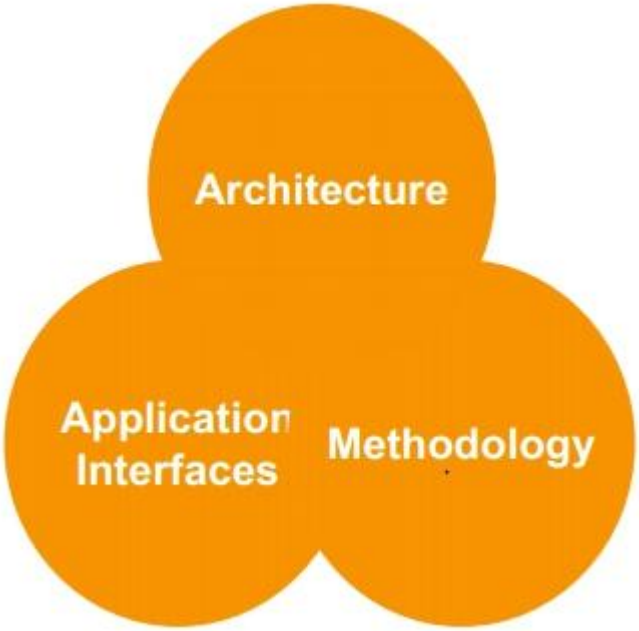
3.2 基本概念

3.3 示例

3.1 方案

Architecture

包含基础软件的架构为硬件无关的应用软件提供集成平台



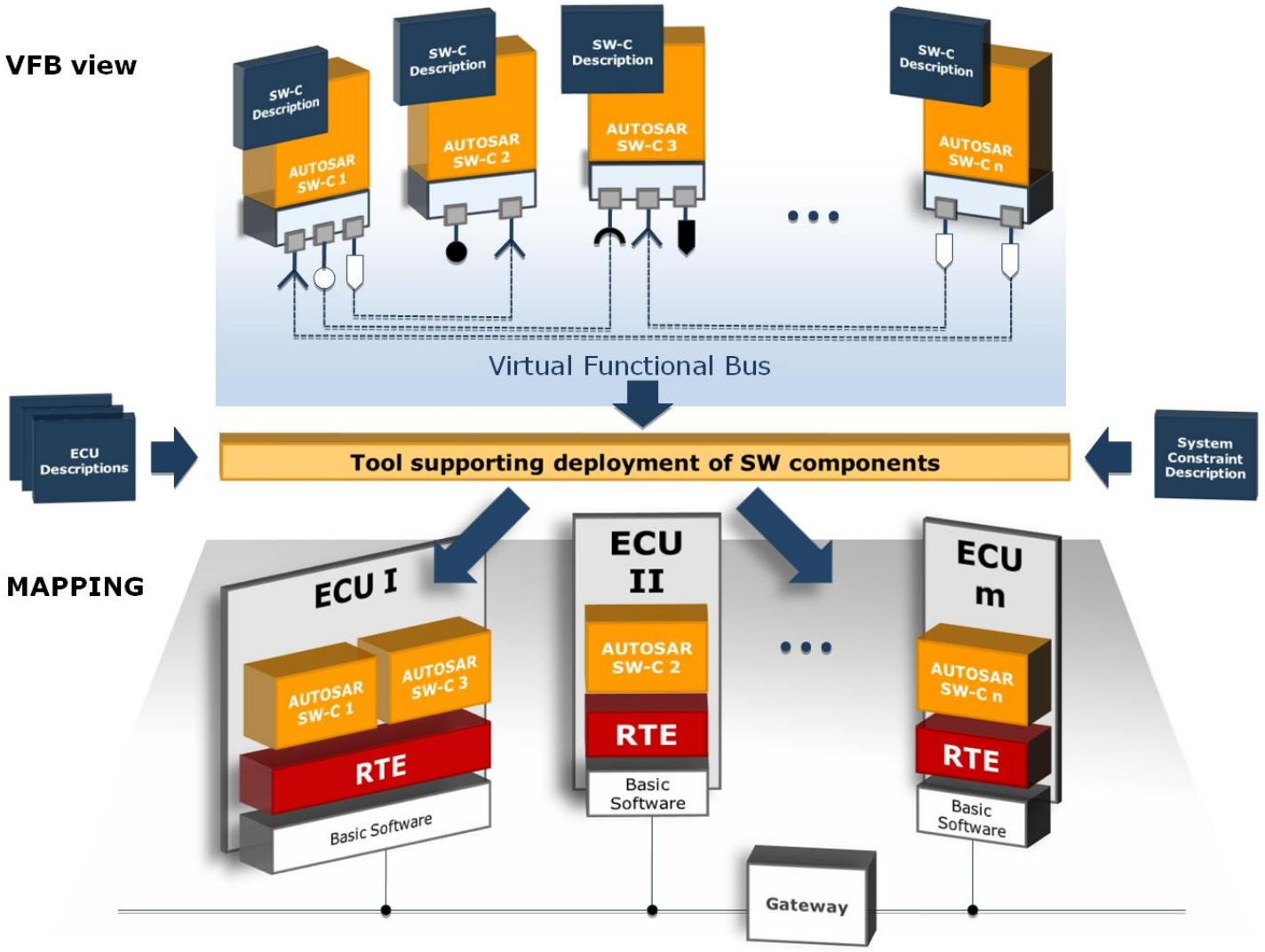
Application Interfaces

规定典型汽车应用的接口

Methodology

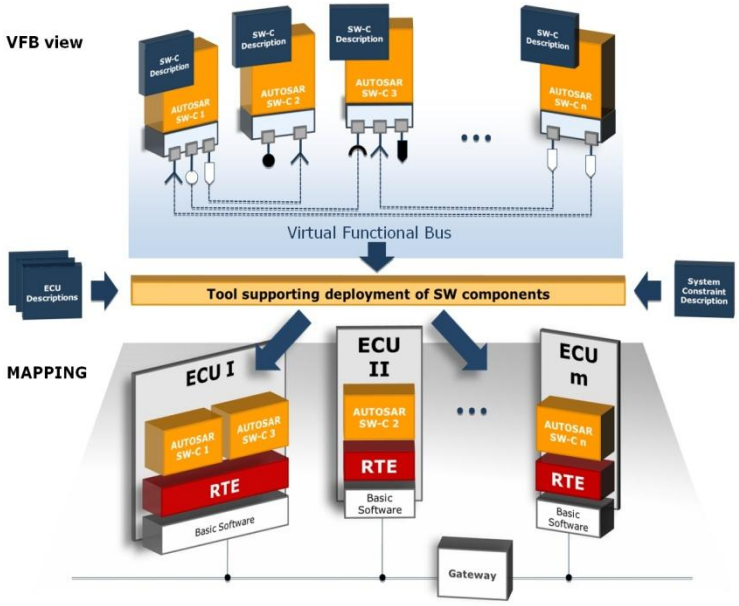
交换格式与描述模板使得配置和集成无缝连接

3.1 方案


VFB view
SW-C Description
AUTOSAR SW-C 1
SW-C Description
AUTOSAR SW-C 2
SW-C Description
AUTOSAR SW-C 3
SW-C Description
AUTOSAR SW-C n
Virtual Functional Bus
ECU Descriptions
Tool supporting deployment of SW components
System Constraint Description
MAPPING
ECU I
AUTOSAR SW-C 1
AUTOSAR SW-C 3
RTE
Basic Software
ECU II
AUTOSAR SW-C 2
RTE
Basic Software
ECU m
AUTOSAR SW-C n
RTE
Basic Software
Gateway

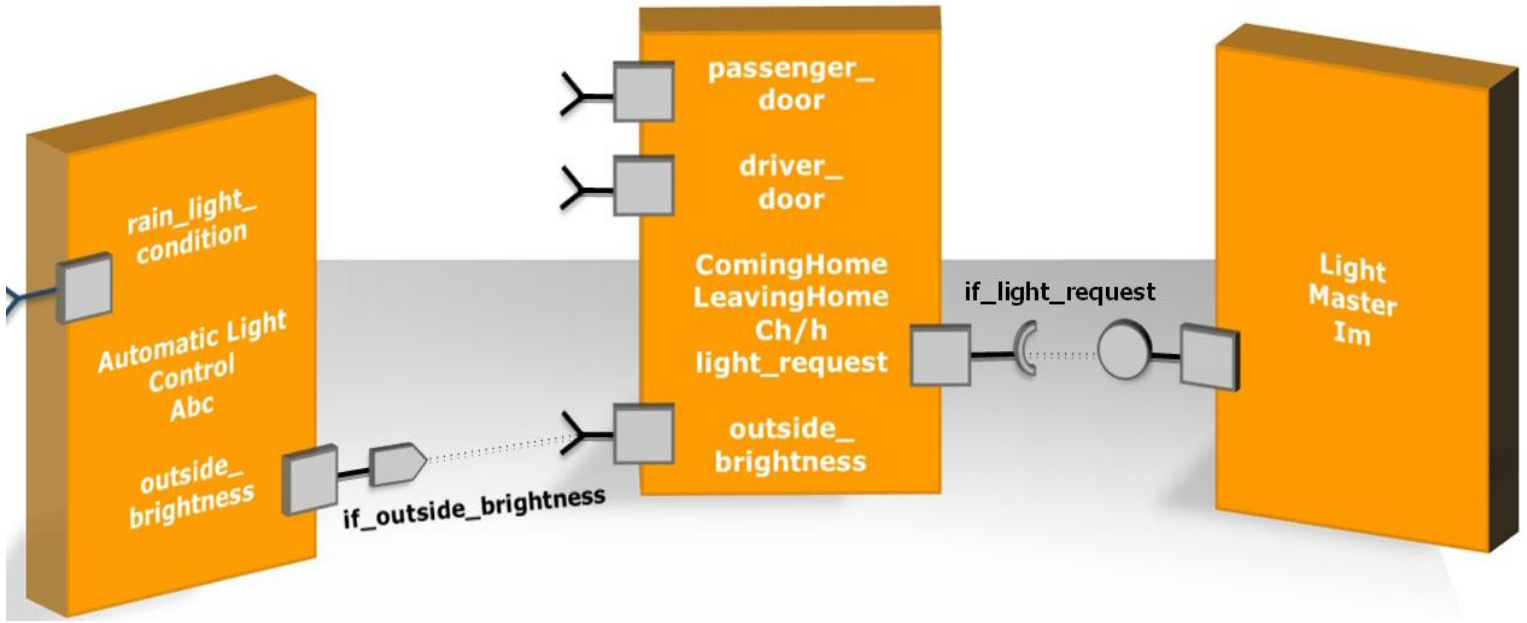
3.2 基本概念

- 软件构件(Software Component)
- 软件构件描述(Software Component description)
- 虚拟功能总线(Virtual Functional Bus)
- 系统约束与ECU描述(System Constraint and ECU Descriptions)
- 映射至ECU
- 运行时环境(RTE)
- 基础软件(Basic Software)



3.2 基本概念

- 软件构件
  - ✓ 构件封装运行于AUTOSAR基础设施上的应用，具有定义良好的接口。



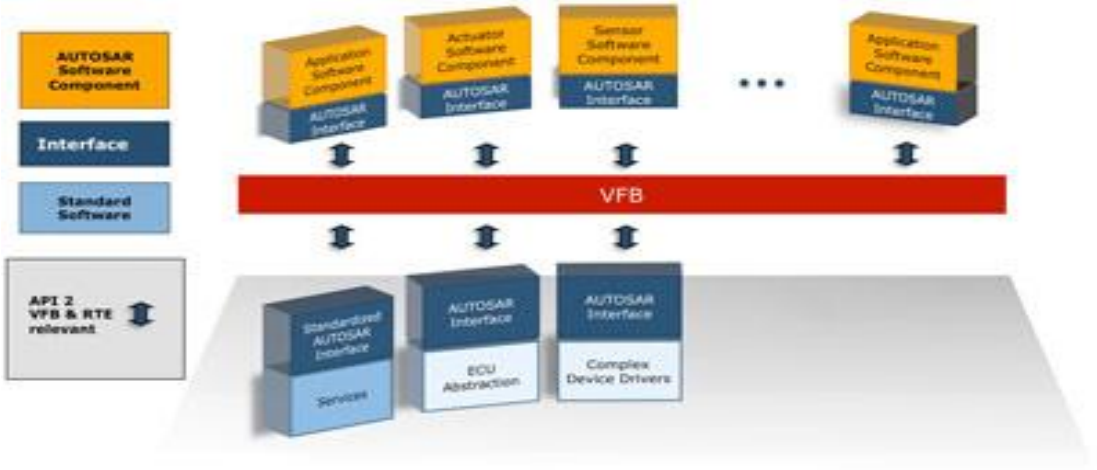
3.2 基本概念

- 软件构件描述
  - 记录软件构件信息以及集成所需的接口信息
  - **AUTOSAR_TPS_SoftwareComponentTemplate.pdf**

```
<COMPOSITION-TYPE>
  <SHORT-NAME>LedControlSystem</SHORT-NAME>
  <COMPONENTS>
    <COMPONENT-PROTOTYPE>
      <SHORT-NAME>BreakSensorPrototype</SHORT-NAME>
      <TYPE-TREF DEST="APPLICATION-SOFTWARE-COMPONENT-TYPE">/ComponentTypes/BreakSensor</TYPE-TREF>
    </COMPONENT-PROTOTYPE>
    <COMPONENT-PROTOTYPE>
      <SHORT-NAME>WarnLightSensorPrototype</SHORT-NAME>
      <TYPE-TREF DEST="APPLICATION-SOFTWARE-COMPONENT-TYPE">/ComponentTypes/WarnLightSensor</TYPE-TREF>
    </COMPONENT-PROTOTYPE>
```

3.2 基本概念

- VFB – 全部通信以及与基础软件的基本接口的集合。当定义好系统软件构件之间的连接，VFB支持在开发的早期阶段进行软件构件的虚拟集成。



3.2 基本概念

- System Constraint and ECU Descriptions
  - 当集成软件构件到ECU网络时，AUTOSAR 提供描述格式描述整个系统(系统约束)以及单个ECU资源与配置的信息(ECU描述)。
  - AUTOSAR_TPS_SystemTemplate.pdf

```
<SYSTEM>
  <SHORT-NAME>system</SHORT-NAME>
  <MAPPING>
    <SHORT-NAME>SystemMapping</SHORT-NAME>
    <DATA-MAPPINGS>
      <SENDER-RECEIVER-TO-SIGNAL-MAPPING>
        <DATA-ELEMENT-IREF>
          <SOFTWARE-COMPOSITION-REF DEST="SOFTWARE-COMPOSITION">/SystemMapping/system/topLevel</SOFTWARE-COMPOSITION-REF>
          <COMPONENT-PROTOTYPE-REF DEST="COMPONENT-PROTOTYPE">/ComponentTypes/LedControlSystem/BreakSensorPrototype</COMPONE
          <PORT-PROTOTYPE-REF DEST="P-PORT-PROTOTYPE">/ComponentTypes/BreakSensor/breakIntensity</PORT-PROTOTYPE-REF>
          <DATA-ELEMENT-REF DEST="DATA-ELEMENT-PROTOTYPE">/InterfaceTypes/BreakInInterface/DE_BreakIn</DATA-ELEMENT-REF>
        </DATA-ELEMENT-IREF>
        <SIGNAL-REF DEST="SYSTEM-SIGNAL">/SystemSignals/syssig_breakin</SIGNAL-REF>
      </SENDER-RECEIVER-TO-SIGNAL-MAPPING>
      <SENDER-RECEIVER-TO-SIGNAL-MAPPING>
        <DATA-ELEMENT-IREF>
          <SOFTWARE-COMPOSITION-REF DEST="SOFTWARE-COMPOSITION">/SystemMapping/system/topLevel</SOFTWARE-COMPOSITION-REF>
          <COMPONENT-PROTOTYPE-REF DEST="COMPONENT-PROTOTYPE">/ComponentTypes/LedControlSystem/WarnLightSensorPrototype</COM
          <PORT-PROTOTYPE-REF DEST="P-PORT-PROTOTYPE">/ComponentTypes/WarnLightSensor/warnLight</PORT-PROTOTYPE-REF>
          <DATA-ELEMENT-REF DEST="DATA-ELEMENT-PROTOTYPE">/InterfaceTypes/WarnLightInterface/DE_WarnLight</DATA-ELEMENT-REF>
        </DATA-ELEMENT-IREF>
```

3.2 基本概念

```
<AR-PACKAGE>
  <SHORT-NAME>Hardware_Topo_acturator</SHORT-NAME>
  <ELEMENTS>
    <ECU-INSTANCE>
      <SHORT-NAME>actrator</SHORT-NAME>
      <CONNECTORS>
        <COMMUNICATION-CONNECTOR>
          <SHORT-NAME>connector_a</SHORT-NAME>
          <CHANNEL-REF DEST="PHYSICAL-CHANNEL">/Hardware_Topo_acturator/canBus_1/physicalChannel_1</CHANNEL-REF>
        </COMMUNICATION-CONNECTOR>
      </CONNECTORS>
    </ECU-INSTANCE>
    <CAN-CLUSTER>
      <SHORT-NAME>canBus_1</SHORT-NAME>
      <PHYSICAL-CHANNELS>
        <PHYSICAL-CHANNEL>
          <SHORT-NAME>physicalChannel_1</SHORT-NAME>
        </PHYSICAL-CHANNEL>
      </PHYSICAL-CHANNELS>
    </CAN-CLUSTER>
  </ELEMENTS>
</AR-PACKAGE>
<AR-PACKAGE>
  <SHORT-NAME>Scalings_acturator</SHORT-NAME>
</AR-PACKAGE>
<AR-PACKAGE>
  <SHORT-NAME>SystemSignals_acturator</SHORT-NAME>
  <ELEMENTS>
    <SYSTEM-SIGNAL>
      <SHORT-NAME>syssig_breakin</SHORT-NAME>
      <LENGTH>32</LENGTH>
    </SYSTEM-SIGNAL>
    <SYSTEM-SIGNAL>
      <SHORT-NAME>syssig_breakout</SHORT-NAME>
      <LENGTH>32</LENGTH>
    </SYSTEM-SIGNAL>
```

3.2 基本概念

- 映射至ECUs
  - AUTOSAR定义方法学和工具支持一个具体ECU系统上各类信息和描述元素的集成，特别是每个ECU上运行时环境和基础软件的配置和生成。
  - 映射即是将软件构件部署到具体的ECU上的过程。

3.2 基本概念

- 运行时环境(RTE)
  - 从AUTOSAR软件构件的角度而言，运行时环境实现了特定ECU上的VFB功能。
- 基础软件
  - 为ECU提供基础设施功能

3.3 示例

- 独立网关项目

主要内容

1. 关于AUTOSAR
2. AUTOSAR标准文档
3. AUTOSAR方案与基本概念
4. AUTOSAR软件架构
5. AUTOSAR方法论

4. AUTOSAR软件架构简介

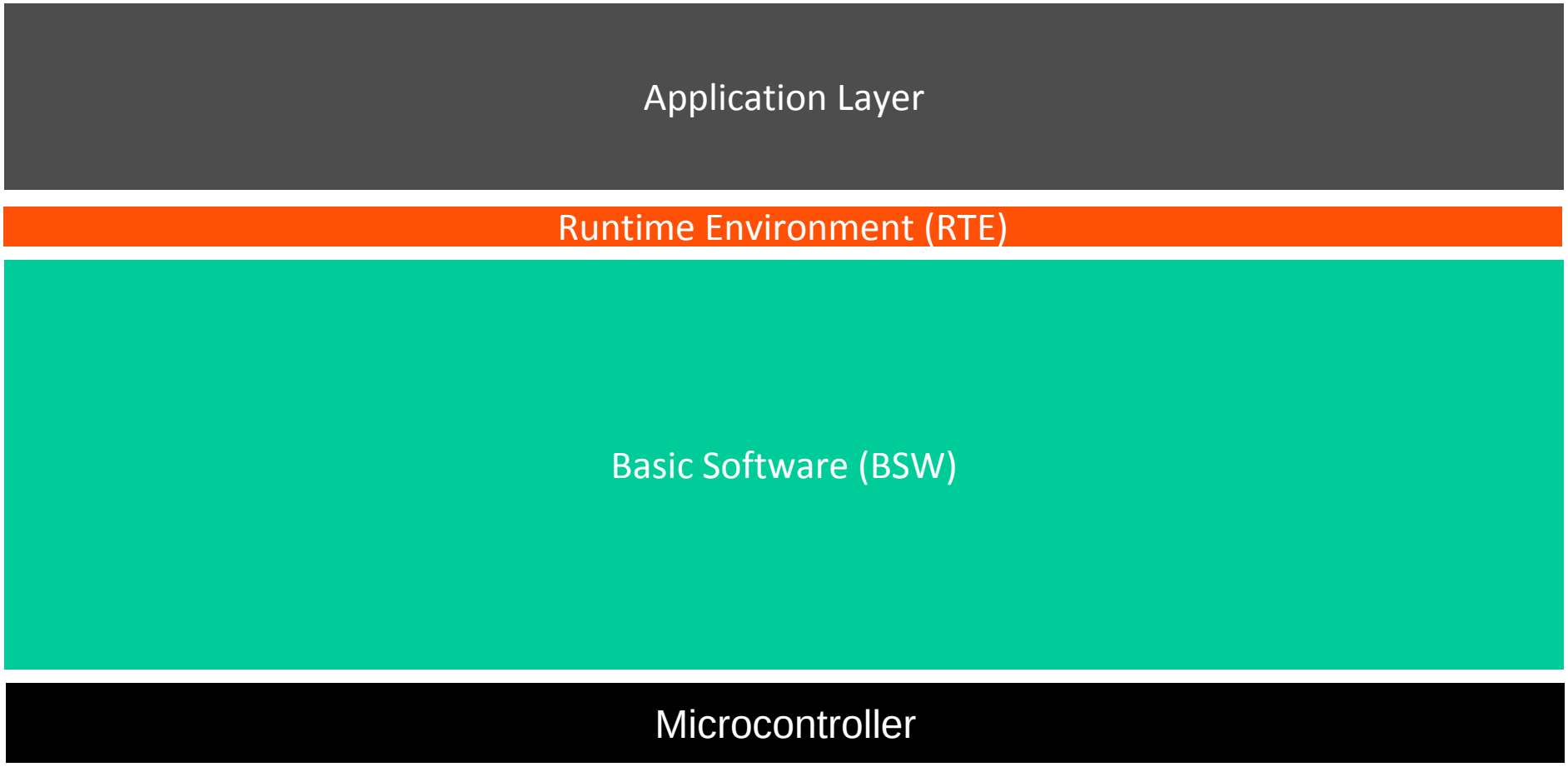
4.1 软件架构视图

4.2 分层简介

4.3 层间交互

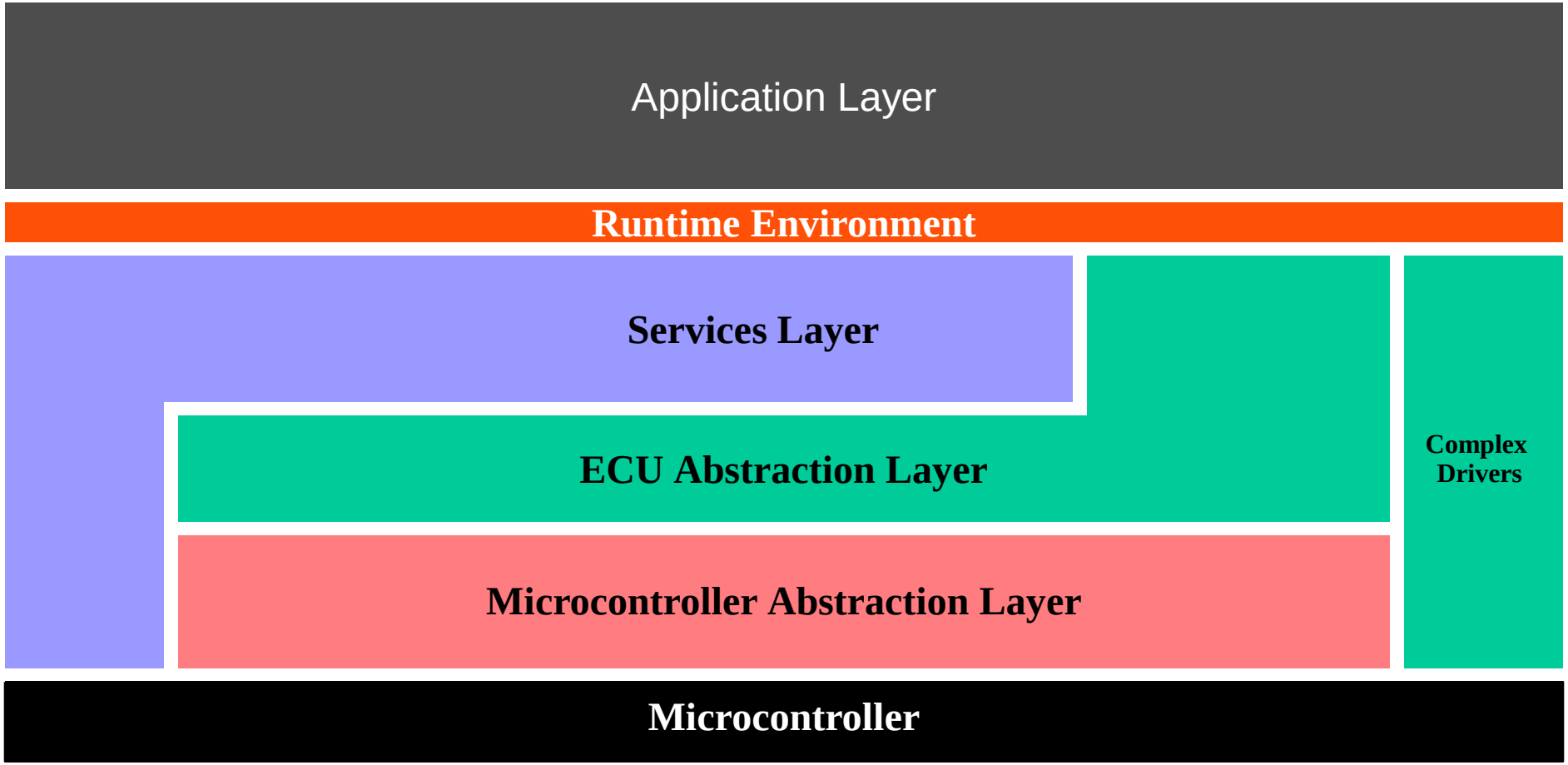
4. 1软件架构视图

- 顶层视图



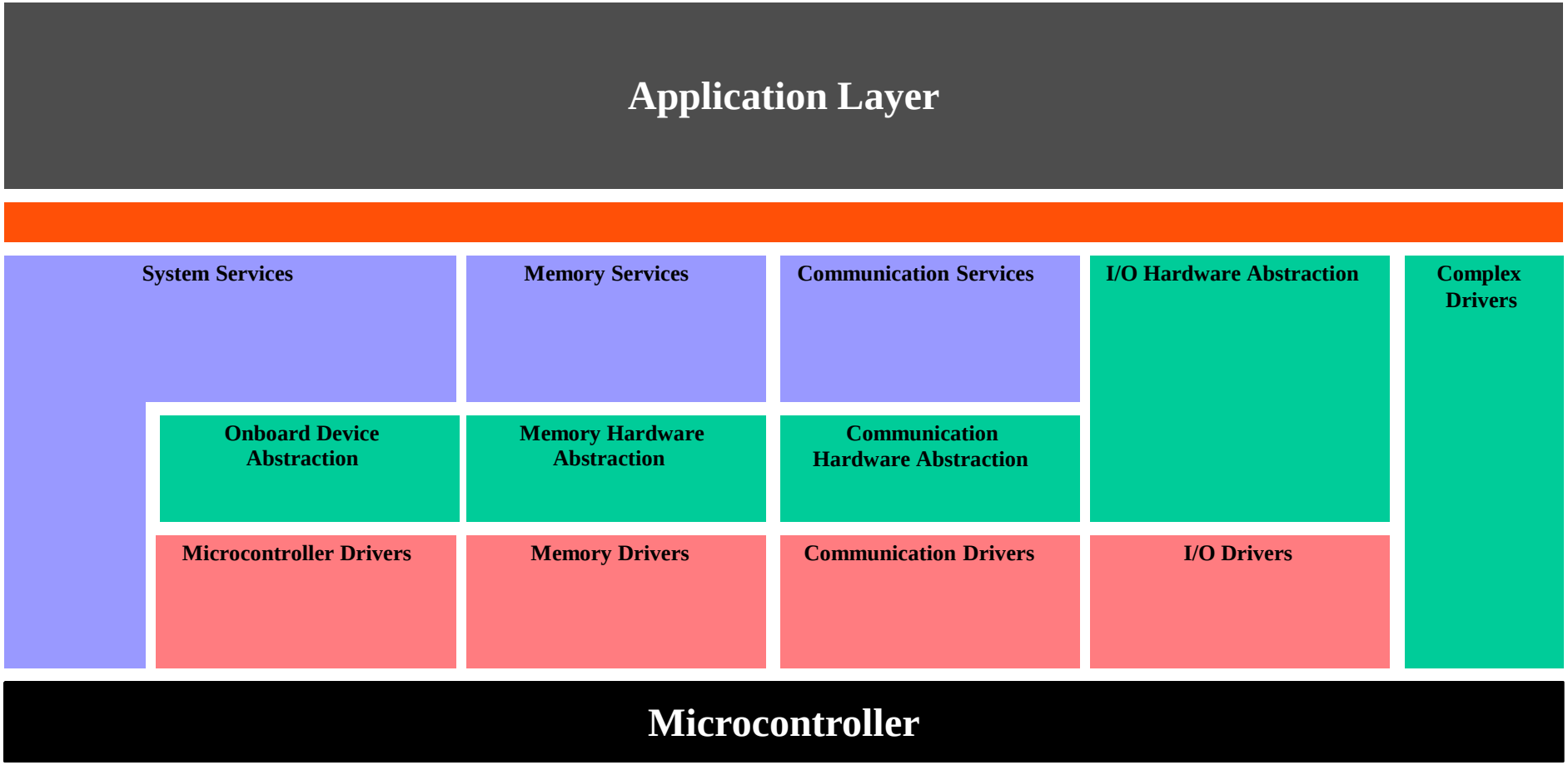
4. 1软件架构视图

- 分层图



4. 1软件架构视图

- 功能组



4. 2分层简介

4.2.1 微控制器抽象层

4.2.2 ECU 抽象层

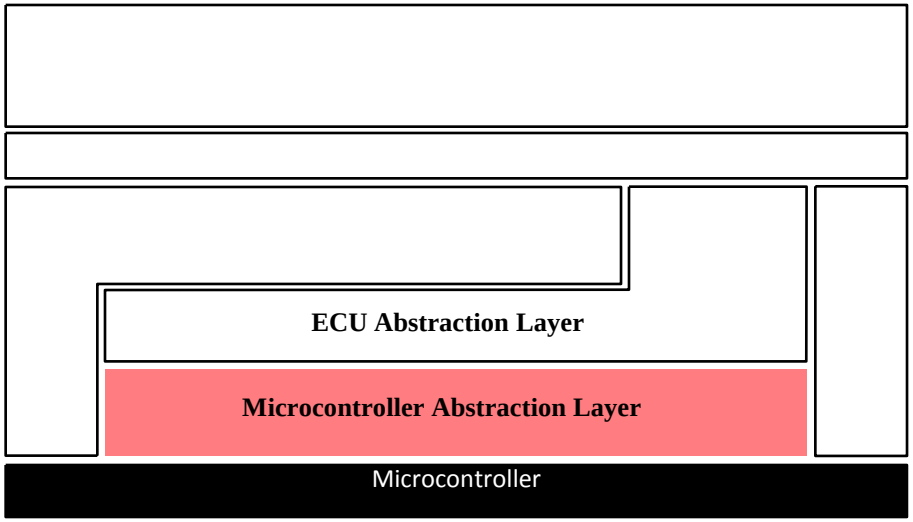
4.2.3 复杂驱动

4.2.4 服务层

4.2.5 AUTOSAR 运行时环境

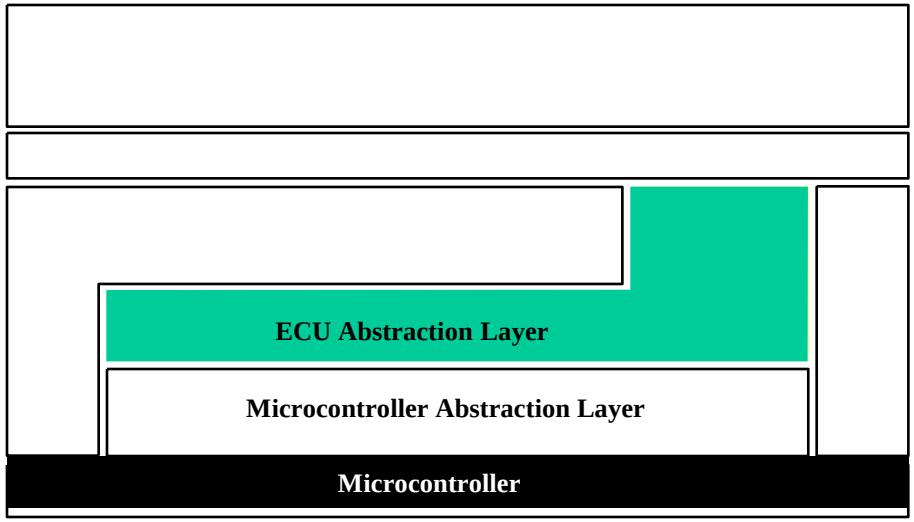
4.2.1 微控制器抽象层

- 包含直接访问微控制器和内部外设的驱动
- 任务 - 使得高层的软件独立于微控制器
- 特性
  - ✓ 实现上依赖于微控制器
  - ✓ 上层接口：标准化、微控制器无关



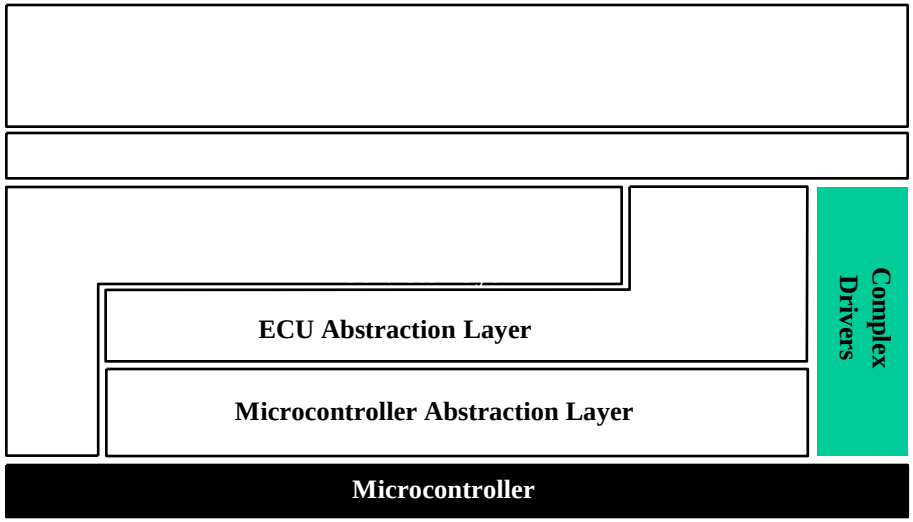
4.2.2 ECU 抽象层

- 包含：微控制器抽象层驱动的接口以及外部设备接口
- 提供访问外设的API（无论设备是在微控制器的内部还是外部，以及如何与微控制器连接）
- 任务：使得高层软件层与ECU硬件布局无关
- 特性：实现上微控制器无关、ECU硬件相关



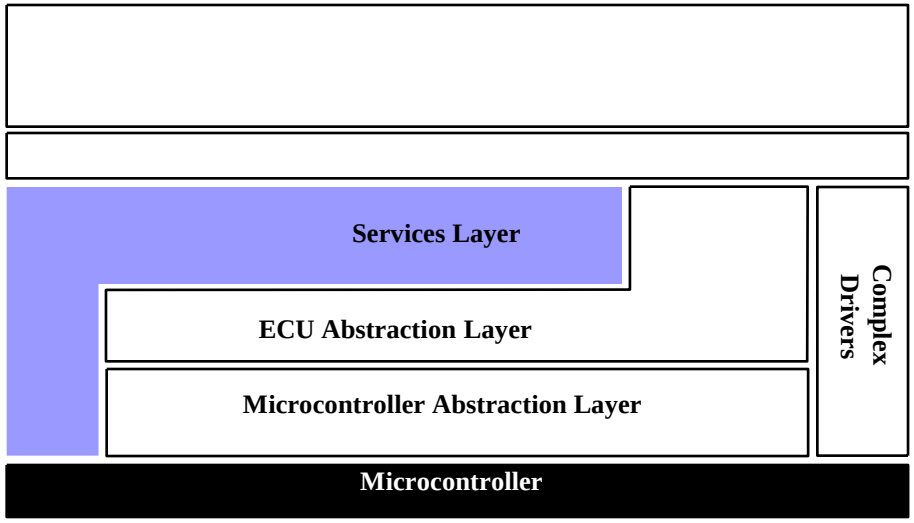
4.2.3 复杂驱动

- 复杂驱动层跨越硬件到运行时环境
- 任务：提供集成特殊功能的能力。例如：AUTOSAR中未规定的设备驱动，具有很高的时间约束或迁移目的。
- 特性：实现上可能依赖于应用、微控制器和ECU硬件；上层接口可能依赖于应用、微控制器和ECU硬件



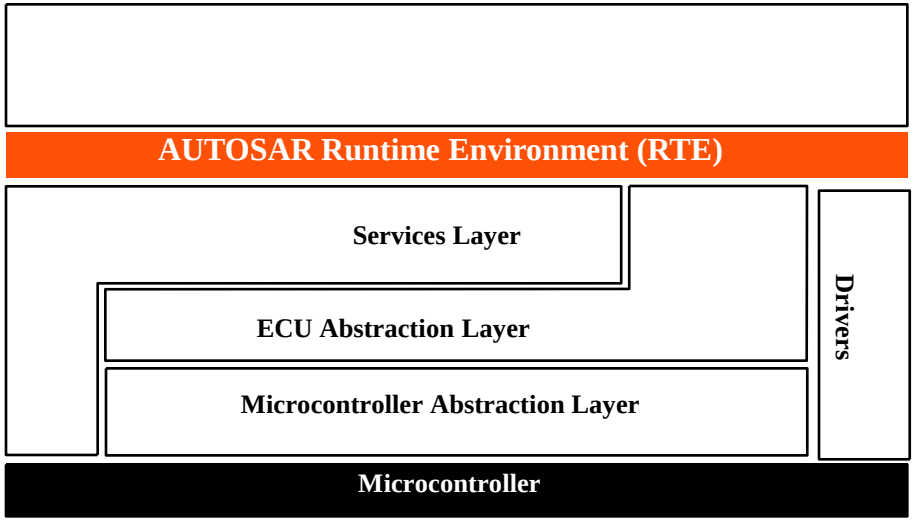
4.2.4 服务层

- 服务层提供操作系统功能、车载网络通信和管理服务、存储服务、诊断服务、ECU状态管理、模式管理、逻辑和时序程序流监测
- 任务：为应用和基础软件模块提供基本服务
- 特性：大部分与微控制器和ECU硬件无关



4.2.5 Runtime Environment

- 运行时环境为应用软件提供通信服务
- 任务：使得AUTOSAR软件构件独立于特定的ECU
- 特性：实现上与ECU和应用相关



4.3 软件层间交互

4.3.1 接口类型

4.3.2 构件与接口视图

4.3.3 接口一般原则

4.3.4 层交互矩阵

4.3.5 复杂驱动接口

4.3.6 示例

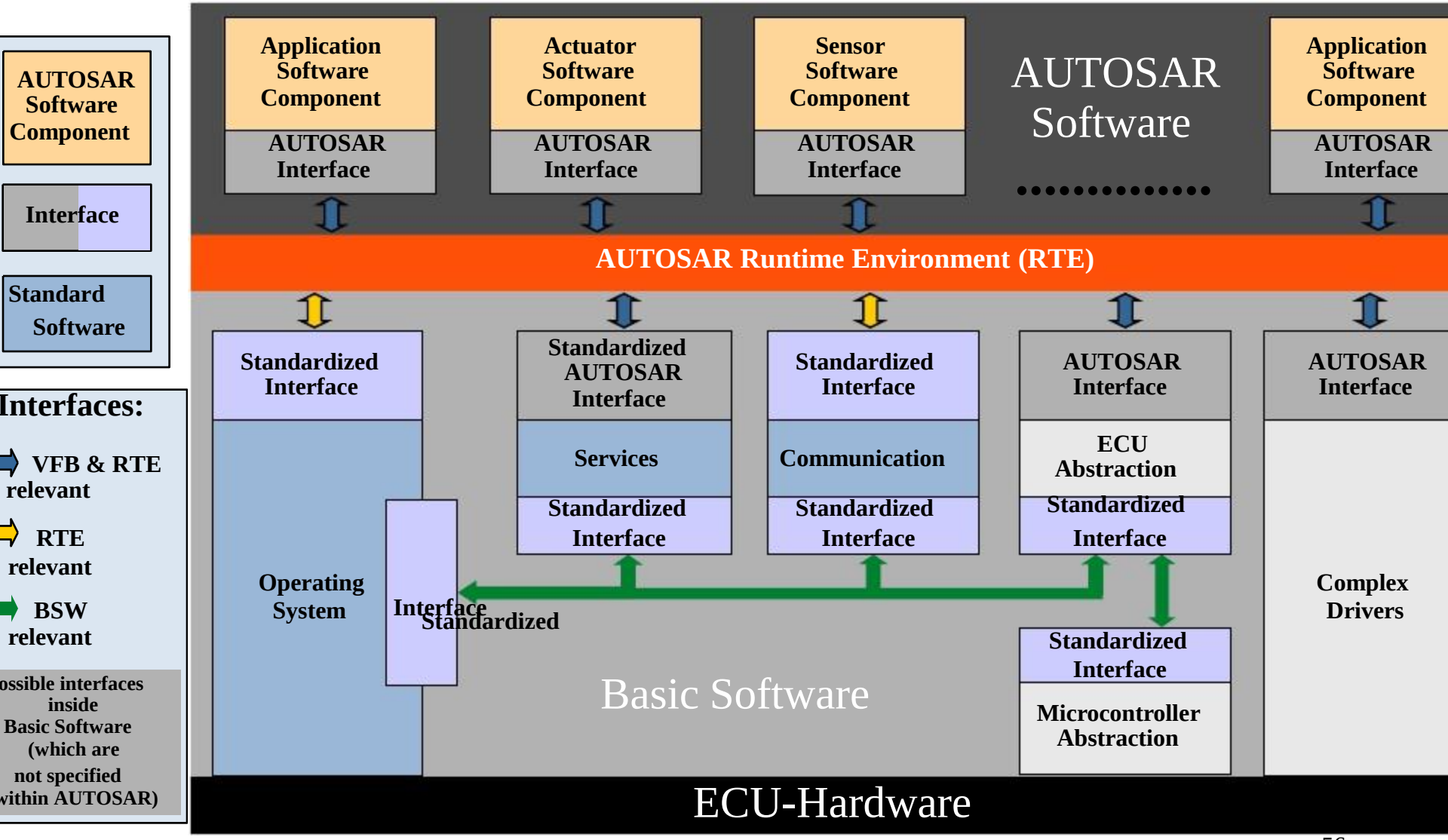
4.4.1 接口类型

- AUTOSAR Interface
- Standardized AUTOSAR Interface
- Standardized Interface

4.3.1 接口类型

接口类型	
AUTOSAR Interface	定义软件构件与/或基础软件模块间交换信息
Standardized AUTOSAR Interface	具有标准化语法与语意的“AUTOSAR Interface”. “Standardized AUTOSAR Interfaces” 典型地用于定义AUTOSAR服务，由AUTOSAR基础软件提供给应用软件构件。
Standardized Interface	没有采用“AUTOSAR Interface” 技术但在AUTOSAR内标准化的 API ，典型地用于相同ECU内软件模块间。

4.3.2构件与接口视图


AUTOSAR Software Component
Interface
Standard Software
Interfaces:
VFB & RTE relevant
RTE relevant
BSW relevant
ossible interfaces inside Basic Software (which are not specified within AUTOSAR)
Application Software Component
AUTOSAR Interface
Actuator Software Component
AUTOSAR Interface
Sensor Software Component
AUTOSAR Interface
AUTOSAR Software
Application Software Component
AUTOSAR Interface
AUTOSAR Runtime Environment (RTE)
Standardized Interface
Operating System
Interface Standardized
Standardized AUTOSAR Interface
Services
Standardized Interface
Standardized Interface
Communication
Standardized Interface
AUTOSAR Interface
ECU Abstraction
Standardized Interface
Standardized Interface
Microcontroller Abstraction
AUTOSAR Interface
Complex Drivers
Basic Software
ECU-Hardware

4.3.3 接口一般原则

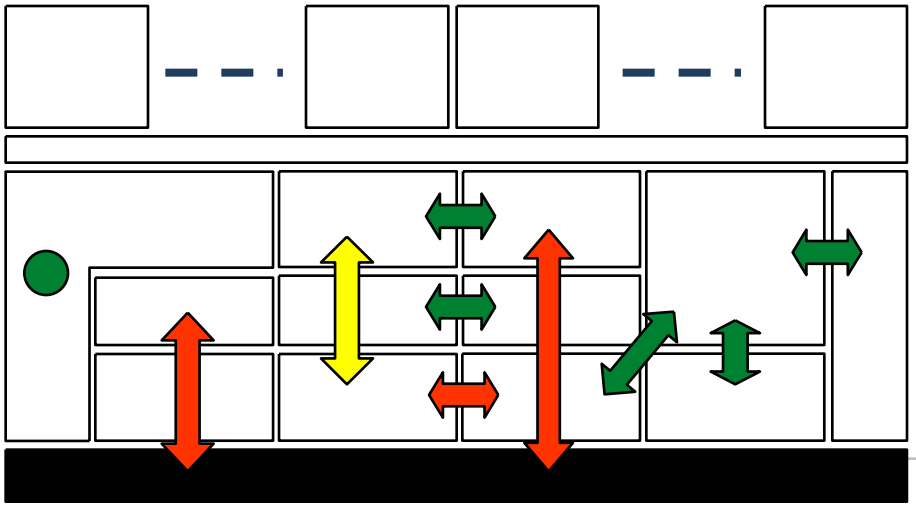
水平接口

服务层：允许水平接口
例如：出错管理器采用NVRAM管理器保存错误数据

ECU抽象层：允许水平接口

复杂驱动可能使用选中的BSW模块

μC 抽象层：不允许接口
特例：允许可配的通知



垂直接口

一层可能访问软件层下的所有接口

应避免绕过一层软件层

不允许绕过两层以上软件层

不允许绕过 μC 抽象层

模块可能访问另一个低层的软件模块
例如 SPI 访问外部硬件

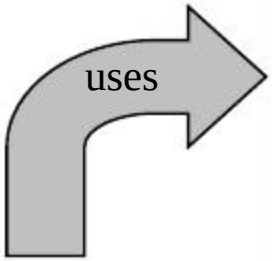
所有层可与系统服务交互

4.3.4 层交互矩阵

× ”is not allowed to use”
Δ “restricted use (callback only)”

The matrix is read **row-wise**:
Example: “I/O Drivers are allowed to use System Services and Hardware, but no other layers”.

(gray background indicates “non-Basic Software” layers)



	System Services	Memory Services	Communication Services	Complex Drivers	I/O Hardware Abstraction	Onboard Device Abstraction	Memory Hardware Abstraction	Communication Hardware Abstraction	Microcontroller Drivers	Memory Drivers	Communication Drivers	I/O Drivers
AUTOSAR SW Components / RTE	✓	✓	✓	✓	✓	×	×	×	×	×	×	
System Services	✓	✓	✓	×	✓	✓	✓	✓	✓	✓	✓	✓
Memory Services	✓	✓	×	×	×	×	✓	×	×	×	×	
Communication Services	✓	✓	✓	×	×	×	×	✓	×	×	×	
Complex Drivers	restricted access -> see the following two slides											
I/O Hardware Abstraction	✓	×	×	×	✓	✓	×	✓	✓	×	✓	✓
Onboard Device Abstraction	✓	×	×	×	×	✓	×	✓	✓	×	✓	✓
Memory Hardware Abstraction	✓	✓	×	×	×	✓	✓	✓	×	✓	✓	×
Communication Hardware Abstraction	✓	×	✓	×	×	✓	×	✓	×	×	✓	✓
Microcontroller Drivers	✓	×	×	×	✓	✓	×	×	Δ	×	×	Δ
Memory Drivers	✓	×	×	×	×	×	✓	×	×	×	×	×
Communication Drivers	✓	×	×	×	×	✓	×	✓	×	×	×	✓
I/O Drivers	✓	×	×	×	✓	✓	×	×	Δ	×	×	Δ

4.3.5 复杂驱动交互

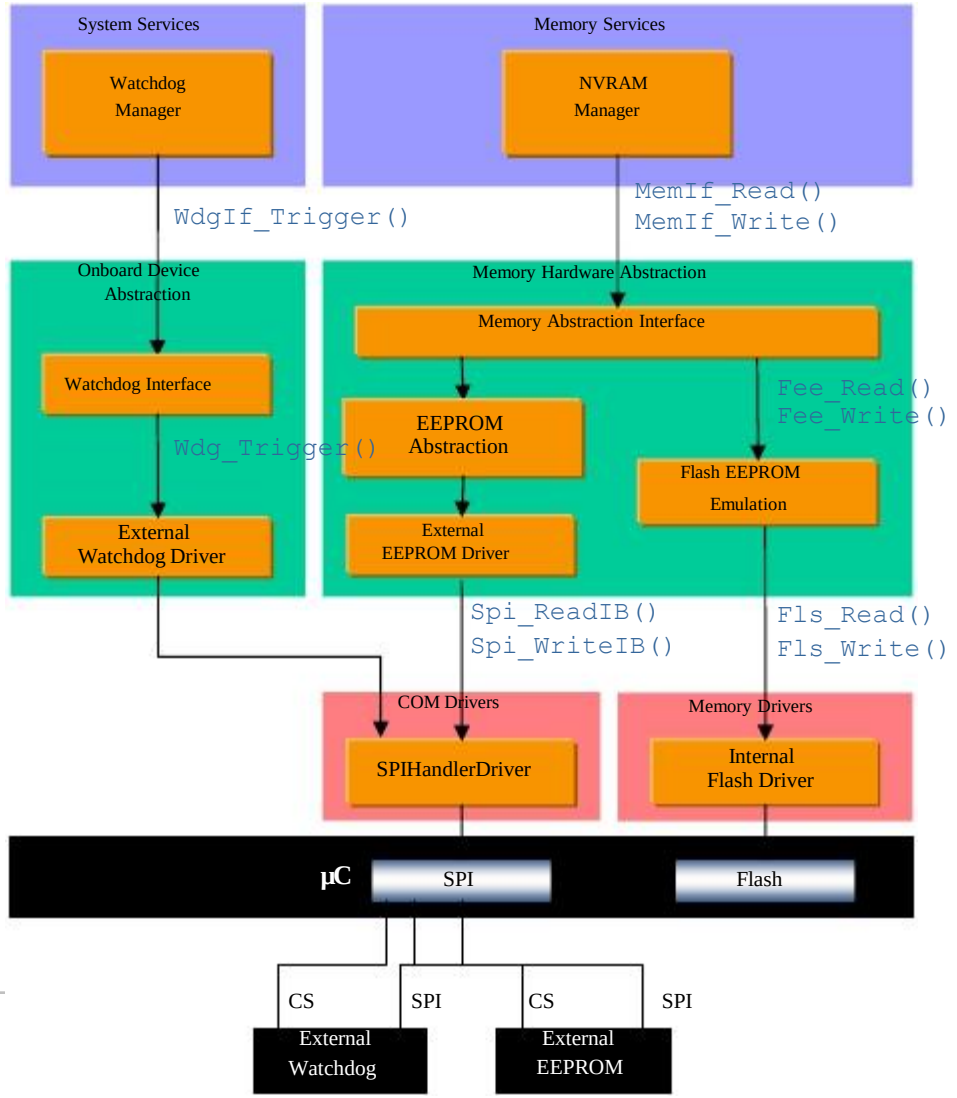
- BSW模块->复杂驱动交互规则
  - ✓如果复杂驱动提供的接口能被访问的模块配置
- 复杂驱动 -> BSW模块交互规则
  - ✓如果基础模块提供接口允许复杂驱动访问，则该接口是可重入的。如果使用回调函数，则名字是可配置的。

4.3.5 复杂驱动交互

- 复杂驱动可能访问的模块
  - ✓ SPI 驱动
  - ✓ GPT 驱动
  - ✓ I/O 驱动
  - ✓ NVRAM 管理器
  - ✓ Watchdog 管理器
  - ✓ PDU 路由器
  - ✓ 总线特定接口模块
  - ✓ NM 接口模块
  - ✓ ECU状态管理器
  - ✓ Det, Dem 和 Dlt
  - ✓ OS

4.3.6 示例-存储

- ECU硬件包含一个外部EEPROM和一个外部看门狗，通过相同SPI连接到微控制器。
- SPIHandlerDriver控制并发访问SPI 硬件，且看门狗访问优先级高于EEPROM访问。
- 微控制器包含一个内部flash
- 存储抽象接口实现方式如下：
  - ✓ 基于设备索引路由
  - ✓ 基于块索引路由
  - ✓ 通过在NVRAM管理器中配置带有功能指针的ROM表（一种虚拟的存储抽象接口）



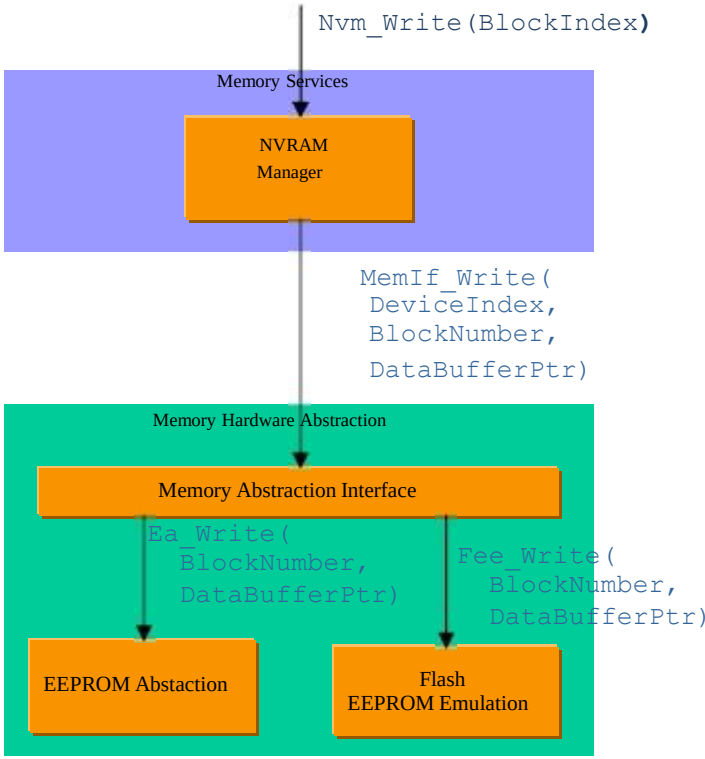
4.3.6 示例

- NVRAM管理器通过存储抽象接口访问驱动。它使用一个设备索引访问不同存储设备
- 存储抽象接口定义

```
Std_ReturnType   MemIf_Write
(
Uint8        DeviceIndex,
Uint16       BlockNumber
Uint8        *DataBufferPtr
)
```

- EEPROM抽象接口定义

```
Std_ReturnType   Ea_Write
(
        uint16                  BlockNumber,
        uint8                   *DataBufferPtr
)
```



4.3.6 示例

- 结论
  - ✓ 抽象层能非常有效地实现
  - ✓ 抽象层可裁剪
  - ✓ 存储抽象接口方便NVRAM管理器访问一个或多个 EEPROM 和 Flash 设备

主要内容

1. 关于AUTOSAR
2. AUTOSAR标准文档结构
3. AUTOSAR方案与基本概念
4. AUTOSAR软件架构
5. AUTOSAR方法论

5 AUTOSAR方法论

5.1 概念

- ✓ 什么是方法学？
- ✓ 什么是方法库？
- ✓ 方法库元素
- ✓ 能力模式

5.2 用例

5.1 概念

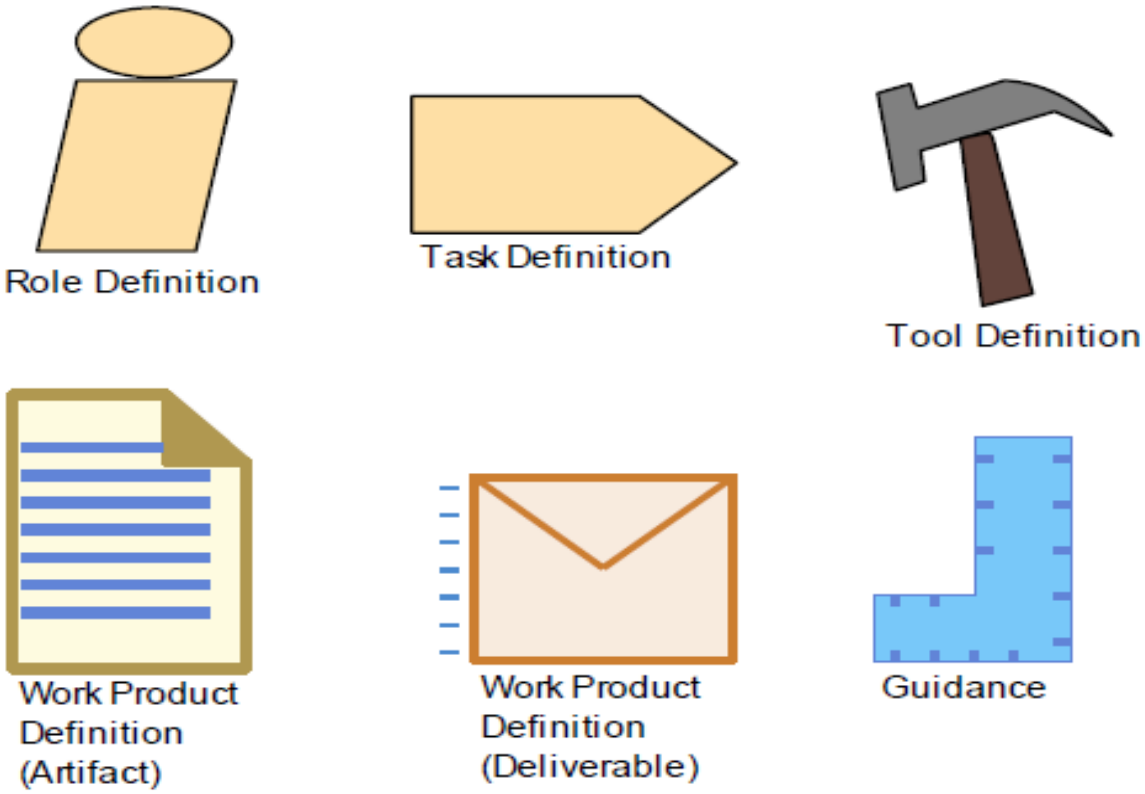
- 什么是方法学？
  - ✓ 方法学以一种通用的、可重用的方法模式，定义了角色执行的、用于产生工作产品的各种活动。
  - ✓ 方法学中的定义与图表均是根据SPEMS来进行描述的，而描述所使用的符号来源于Enterprise Architect modeling tool。

5.1 概念

- 什么是方法库？
  - 定义与描述了每个方法模式（method pattern）中的方法库元素，例如角色（roles），任务（tasks），工作产品定义（work product definitions）。
- 方法库元素
  - 任务定义（Task Definition）
  - 工作产品定义（Work Product Definition）
  - 角色定义（Role Definition）
  - 工具定义（Tool Definition）
  - 指南定义（Guidance）

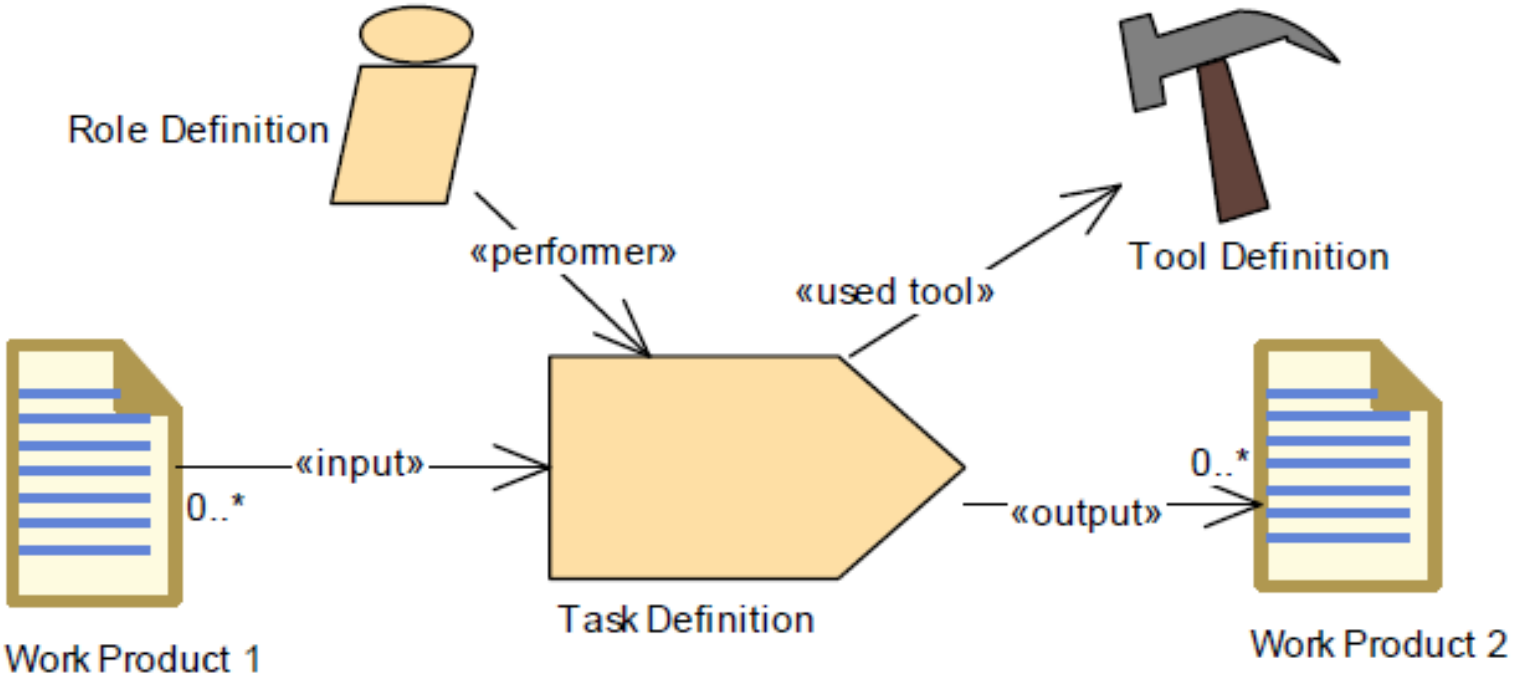
5.1 概念

- 方法库元素符号



5.1 概念

- 示例


Role Definition
«performer»
«used tool»
Tool Definition
«input»
0..*
Task Definition
«output»
0..*
Work Product 1
Work Product 2

5.1 概念

- 什么是任务定义？

  根据SPEM 元模型，一个任务定义是指将被特定角色执行的、可分配的工作单元。一个任务的持续期大致为几个小时到几天，通常会生成一个或者多个工作产品。任务通常与多个输入输出工作产品相关联，且强制性的输入与可选择性的输入是可区分的。

- 特性
  - ✓ 在AUTOSAR方法学中，任务是可重用的元素
  - ✓ 至少与一个执行角色相关联
  - ✓ 通过工具来获得输出结果

5.1 概念

- 什么是工作产品？

  根据SPEM元模型，工作产品定义是由任务使用、修改和生成的。

- 特性
  - ✓ 作为定义可重用资源的基础服务；
  - ✓ 可以通过嵌套关系相互关联。

5.1 概念

- Artifact

一种实体的工作产品，被一个或者多个任务所消耗、生成和修改。一个Artifact可以为其他Artifact的某个组成部分，也可以作为定义可重用资源的基础服务。

- Artifact的类型
  - ✓ AUTOSAR XML
  - ✓ Source code
  - ✓ Binary code(executable)
  - ✓ Text

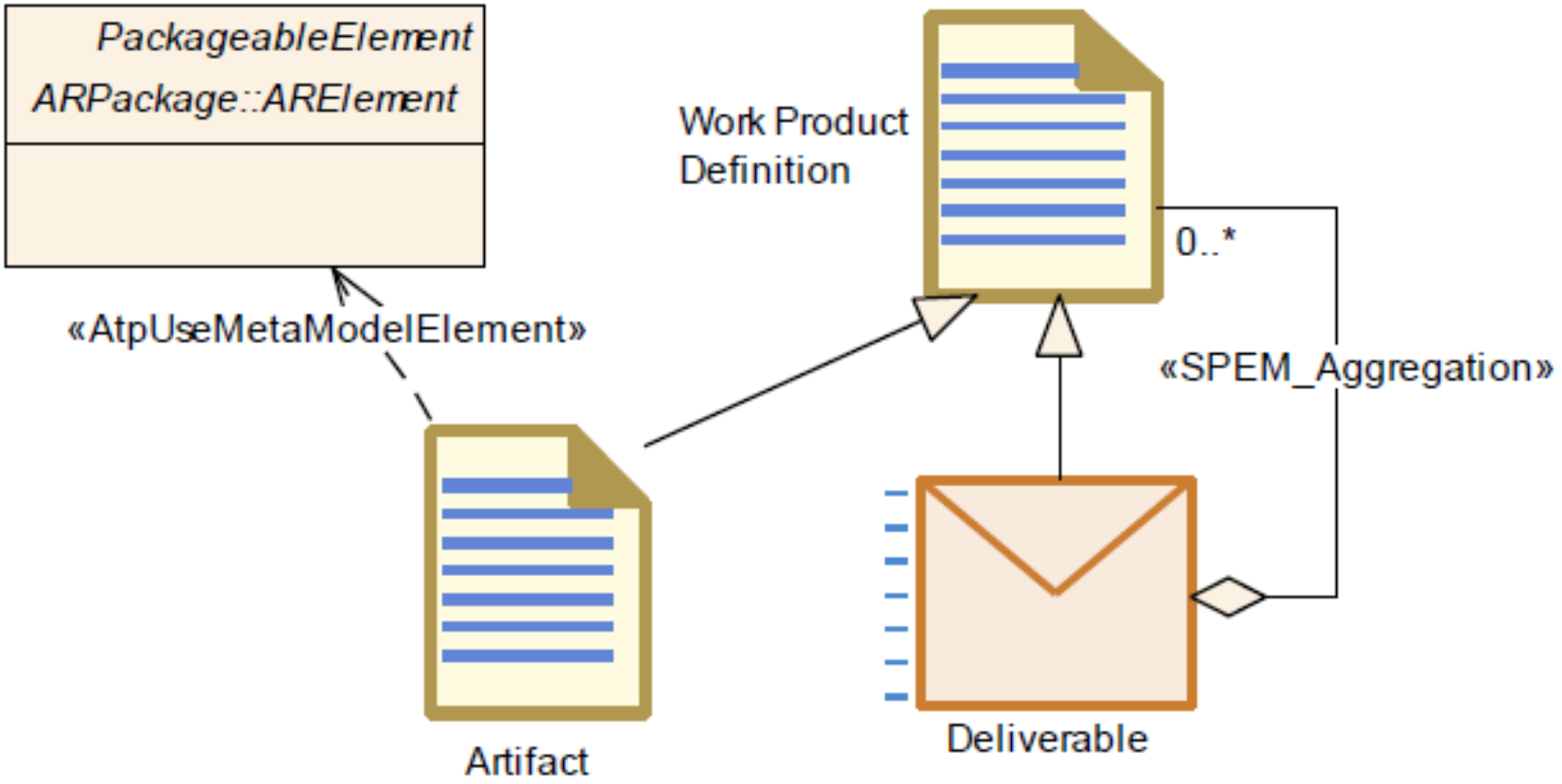
5.1 概念

- Deliverable

用于预定义那些需打包交付的、典型的、值得推荐的内容，而这些内容通常以工作产品的形式来描述。可交付的内容通常是开发过程中对客户或者开发人员有使用价值的输出资源，是整合了其他工作产品的工作产品聚合物。在AUTOSAR方法学中，聚合关系指示了一个可交付实体包含的工作产品内容。

5.1 概念

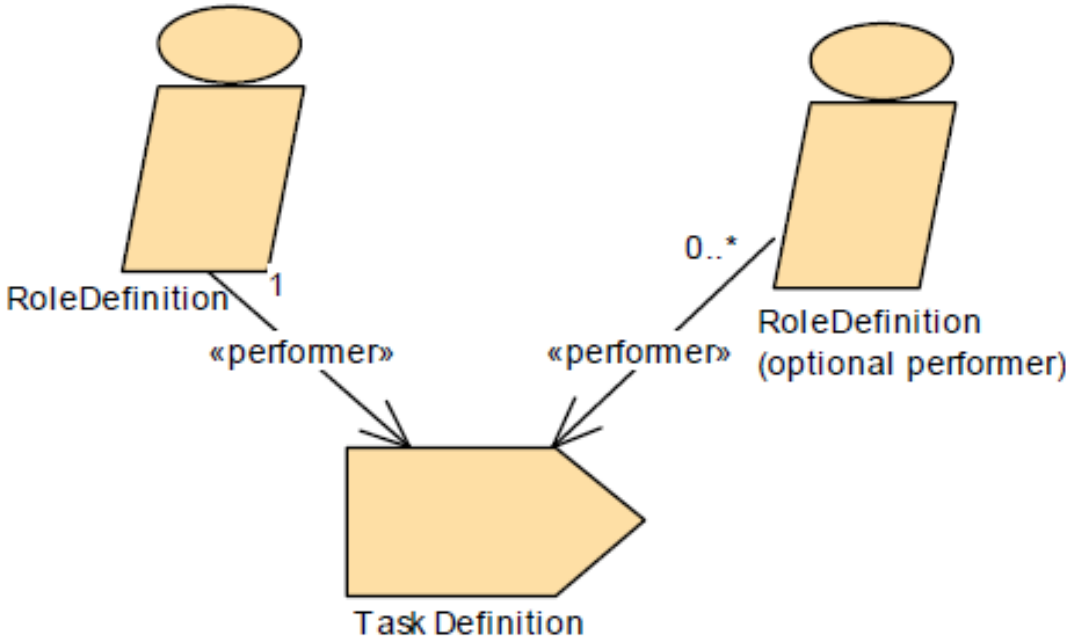
- 示例


PackageableElement
ARPackage::ARElement
«AtpUseMetaModelElement»
Work Product
Definition
0..*
«SPEM_Aggregation»
Artifact
Deliverable

5.1 概念

- 什么是角色定义？

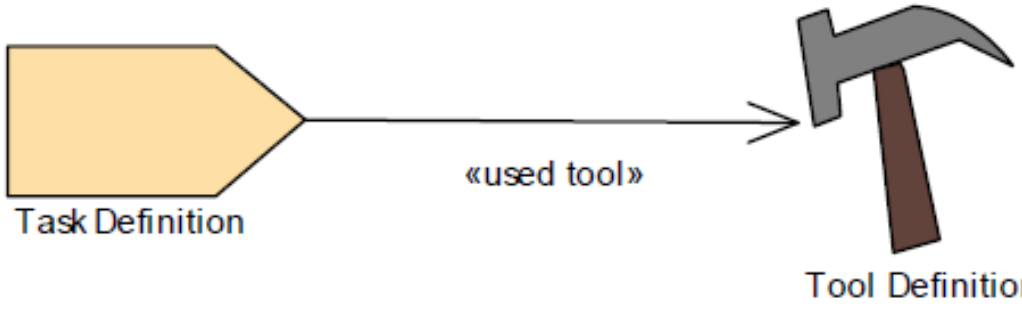
角色定义了个体或者个体集合的职责，以及他们执行一个任务所需要的技术、能力以及资格。一个角色可能被一个人或者多个人承担，而一个人可能充当几个角色，每个角色执行多个任务。



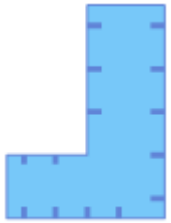
5.1 概念

- 什么是工具定义？

用于说明某种工具是如何参与任务执行的，工具定义描述了相关角色在执行任务工作时，其使用的支持工具的功能。工具可以为任务的完成，确定某个资源为有用、必需、可推荐的资源，也可以用来管理一个或者多个工作产品。



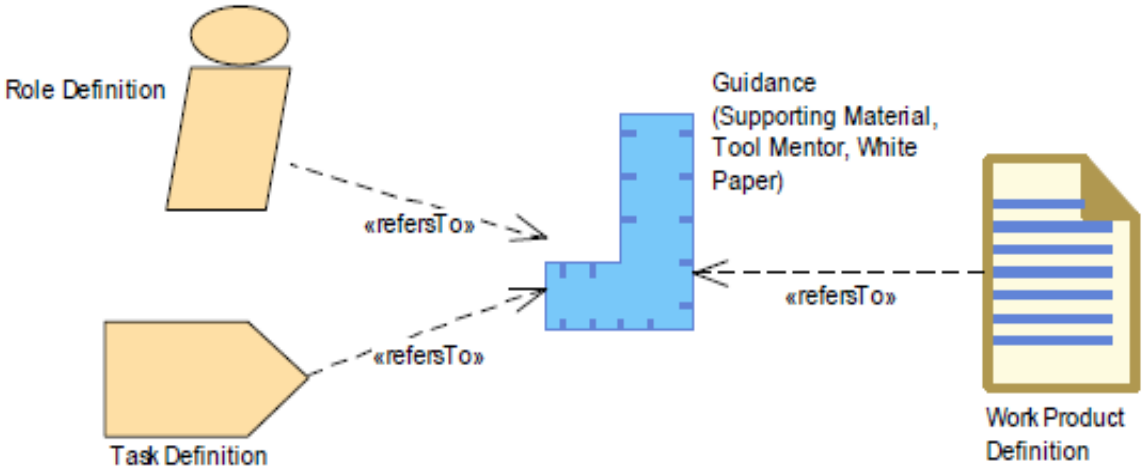
5.1 概念



- 什么是指南？

  提供了与角色、任务、工作产品相关的额外信息。

- 类型
  - ✓ 支持材料
  - ✓ 工具指导书
  - ✓ 白皮书



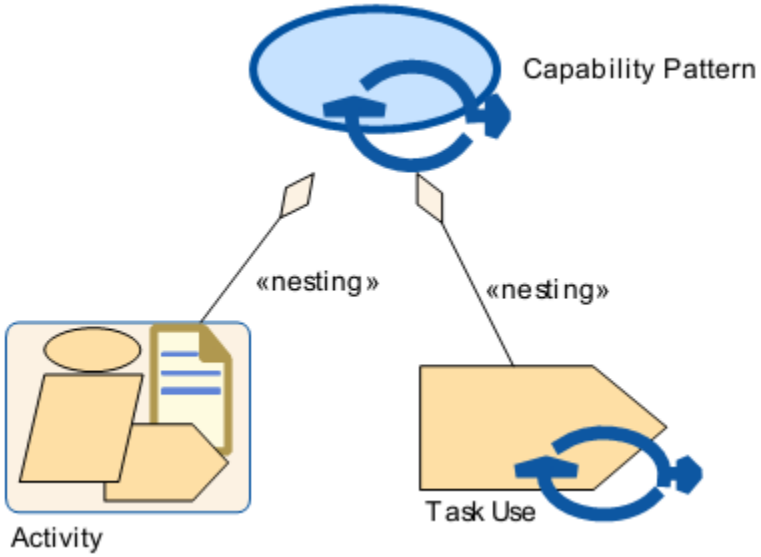
5.1 概念

- 什么是能力模式？ Capacity Pattern

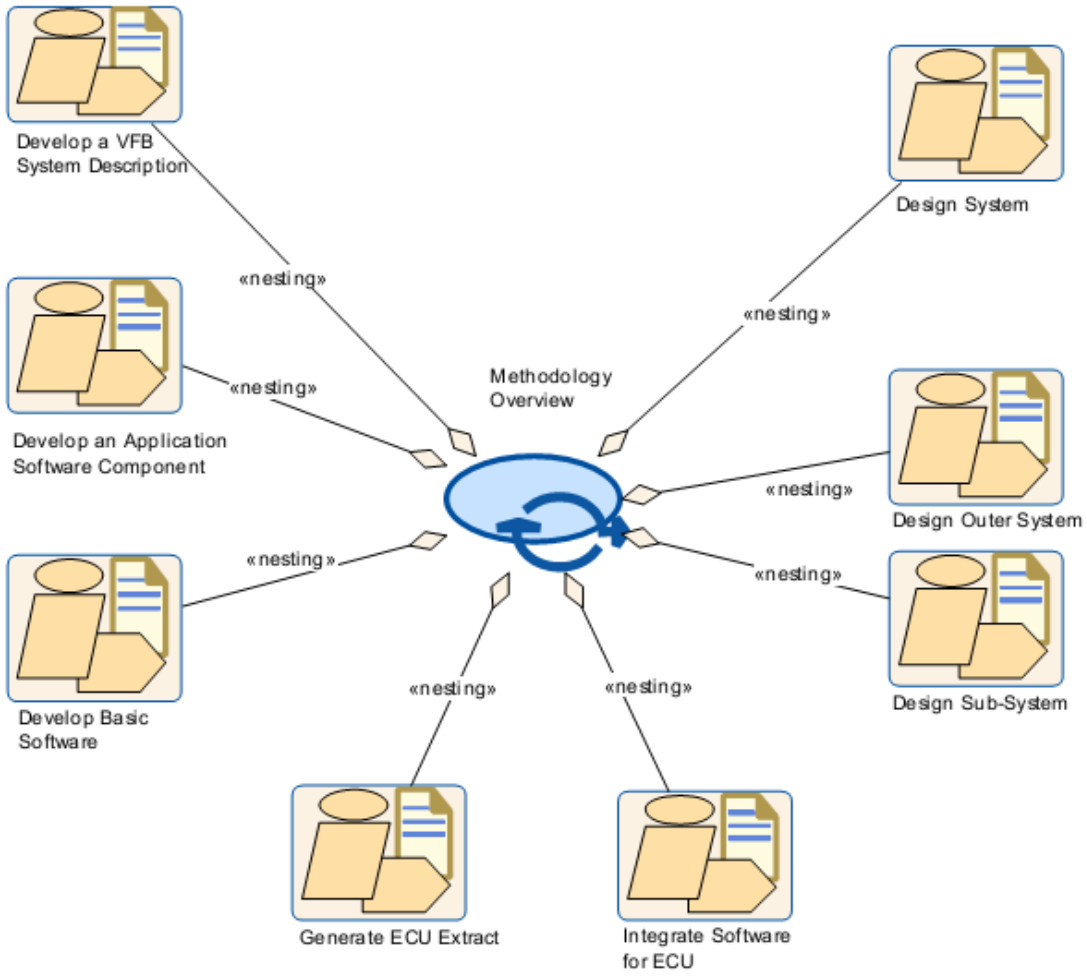
是一个包含一套可重用活动的过程模式。可组装成更大的能力模式,用于描述开发过程或部分开发过程，包括典型的用例。

- 用例元素
  - ✓ 任务（Task Use）--代表一个特定活动背景下的任务定义。
  - ✓ 活动（Activity）--定义过程的主要构建块，是工作分解元素的集合。
  - ✓ 能力模式（Capability Pattern）

5.1 概念



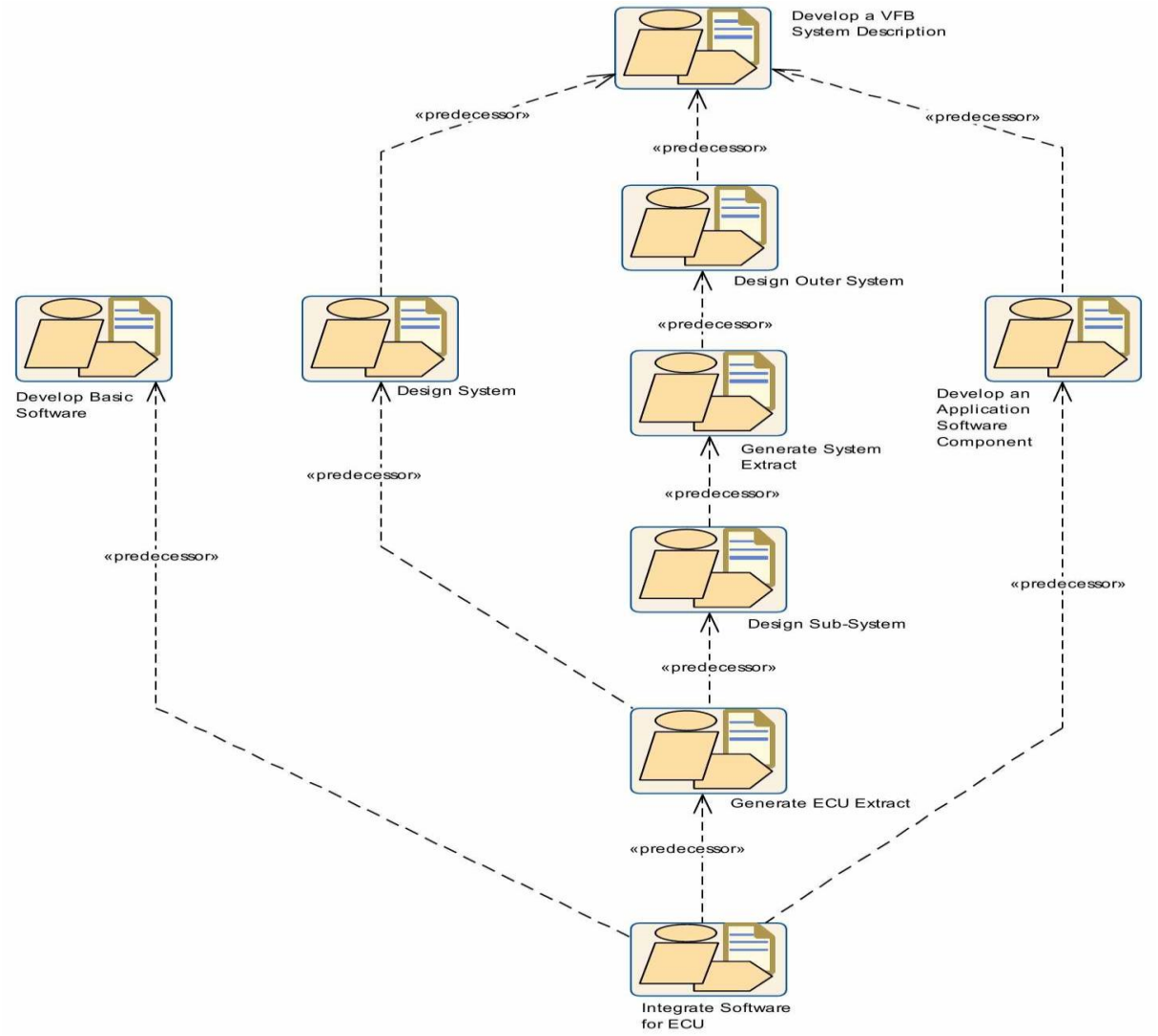
5.2 用例


Develop a VFB System Description
Design System
«nesting»
«nesting»
Develop an Application Software Component
Methodology Overview
«nesting»
«nesting»
Design Outer System
«nesting»
«nesting»
Develop Basic Software
Design Sub-System
«nesting»
«nesting»
Generate ECU Extract
Integrate Software for ECU

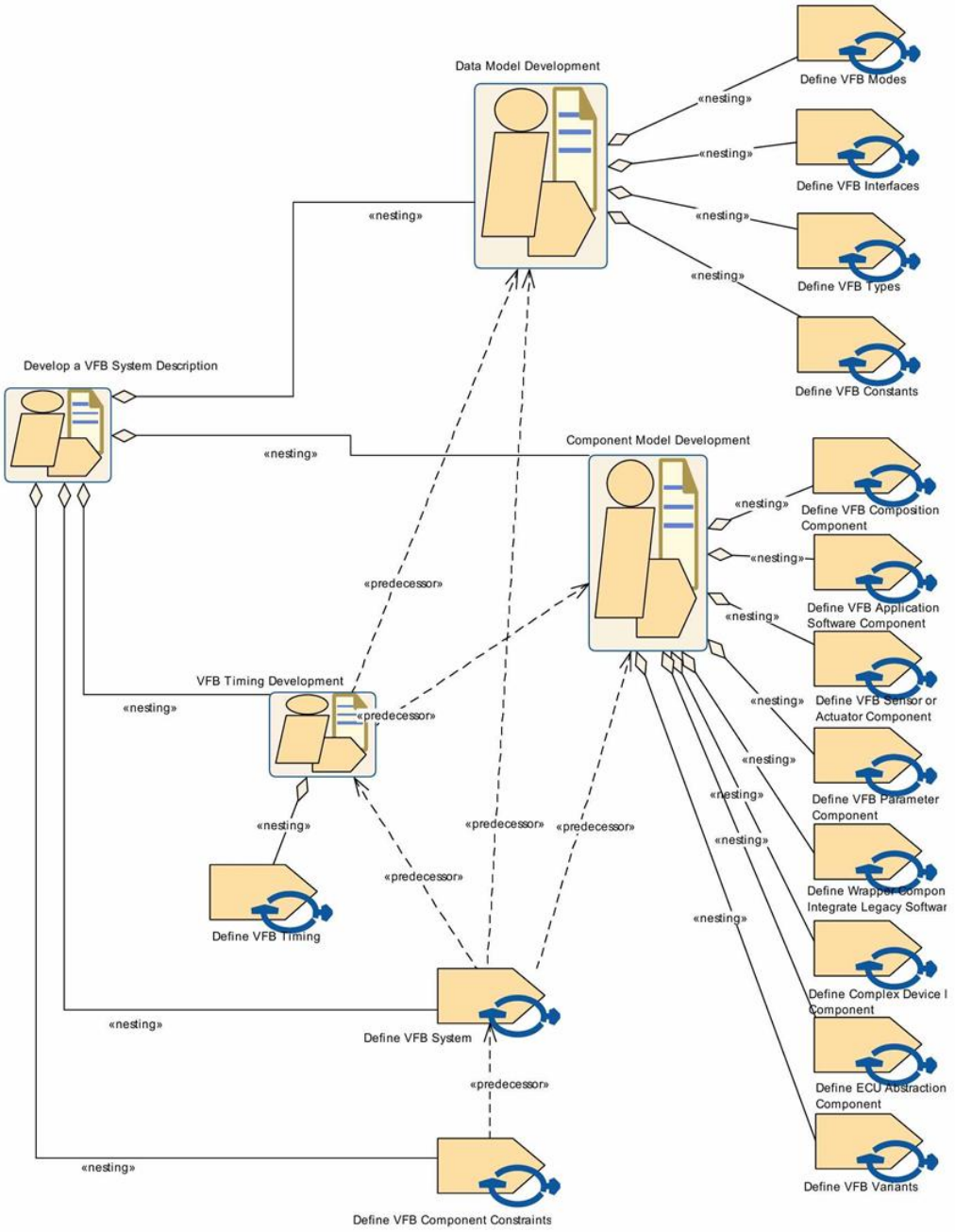
5.2 用例

- Develop VFB
- Develop Software Component
- Develop System
- Develop Basic Software
- Integrate Software for ECU
- Component and Service
- Calibration
- Memory Mapping

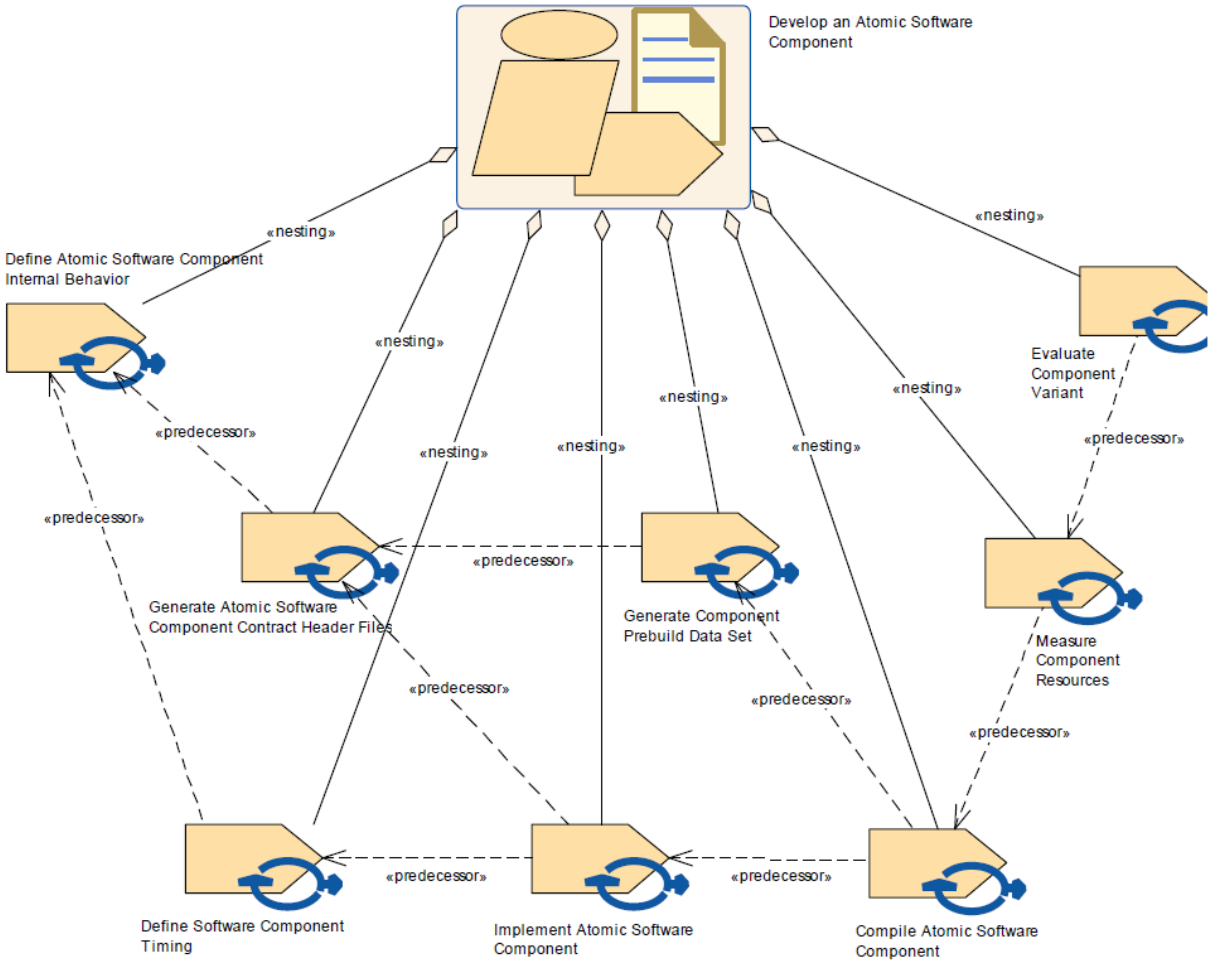
5.2 用例


Develop a VFB System Description
«predecessor»
«predecessor»
«predecessor»
Design Outer System
«predecessor»
Develop Basic Software
Design System
Generate System Extract
Develop an Application Software Component
«predecessor»
«predecessor»
«predecessor»
«predecessor»
Design Sub-System
«predecessor»
«predecessor»
Generate ECU Extract
«predecessor»
Integrate Software for ECU

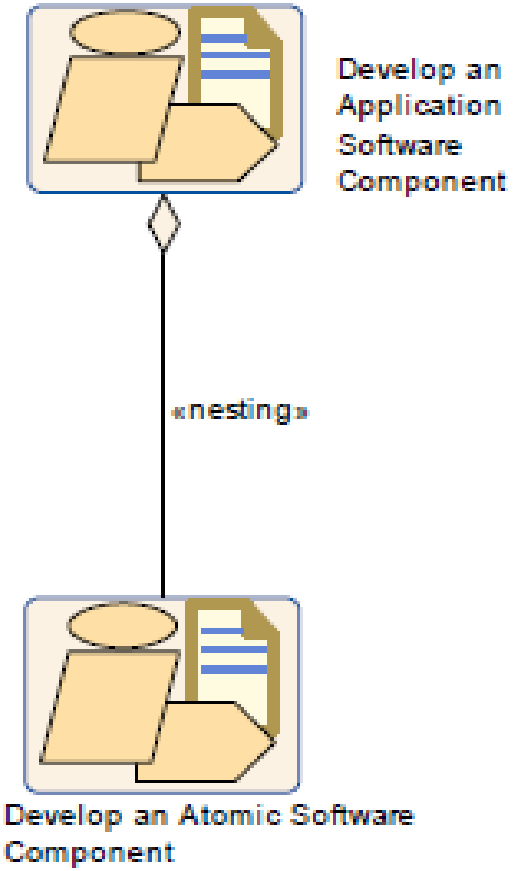
5.2.1 Develop VFB



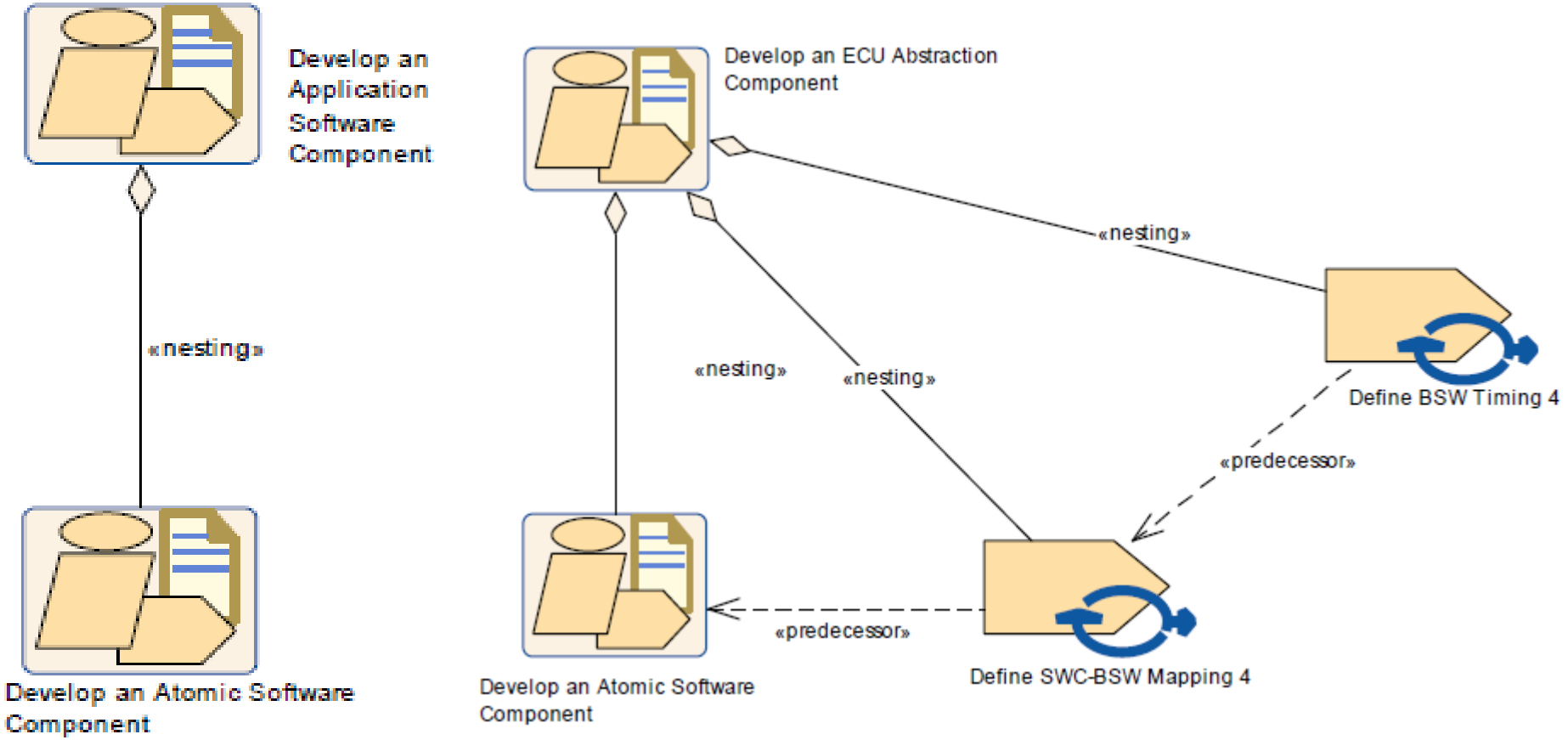
5.2.2 Develop Atomic SWC



5.2.3 Develop Application SWC

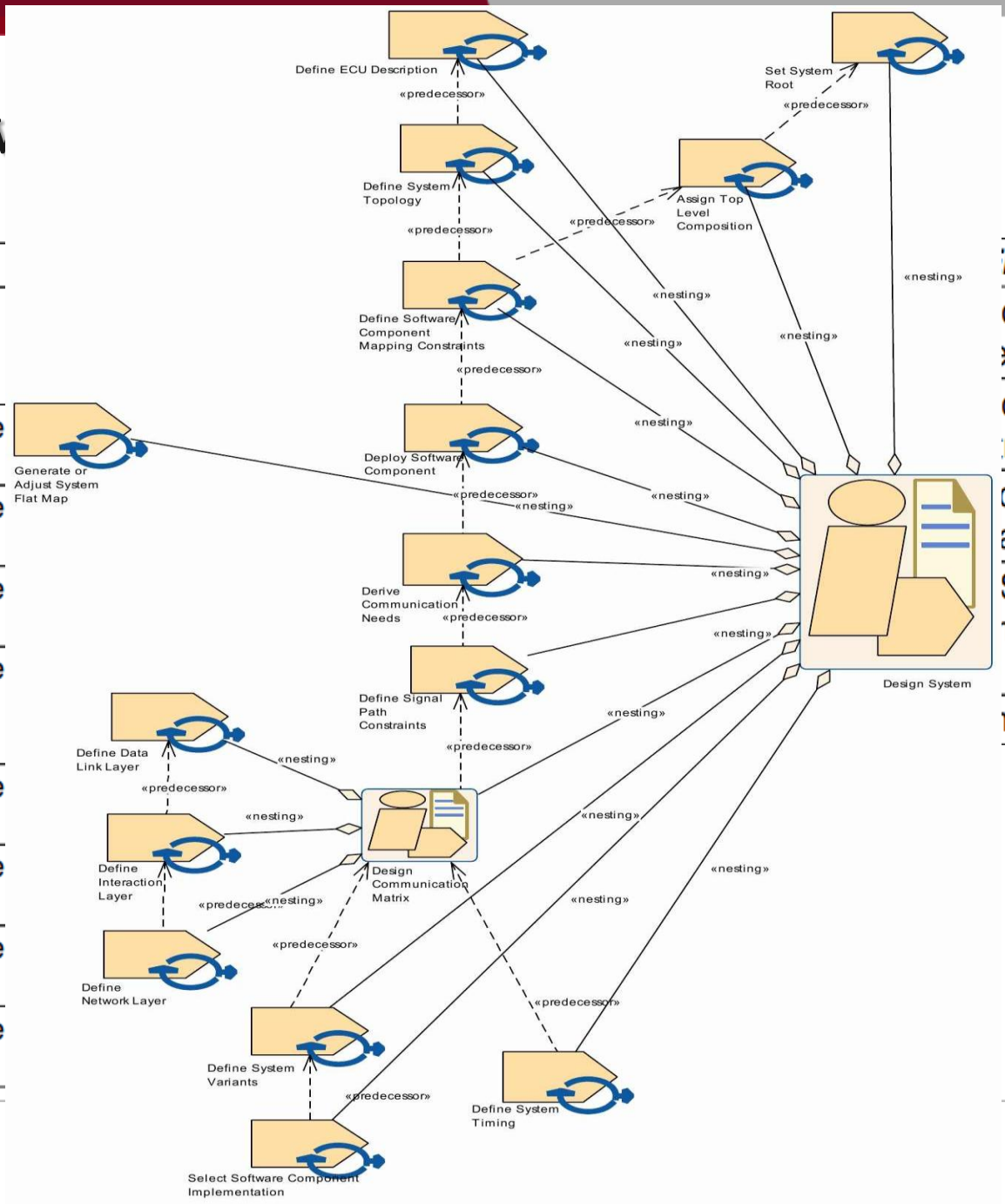


5.2.4 Develop Specialized SWC

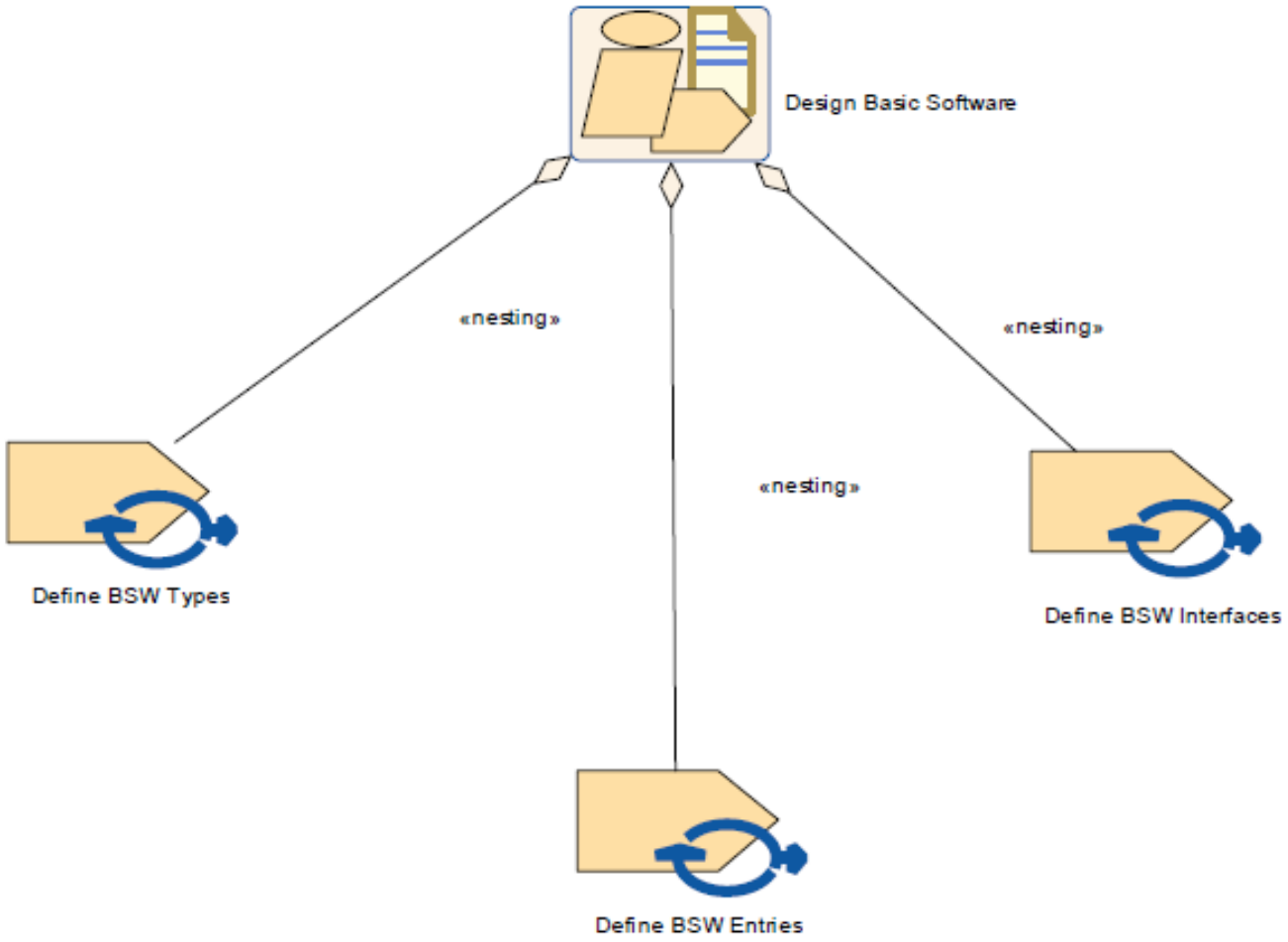


5.2.5 Dev

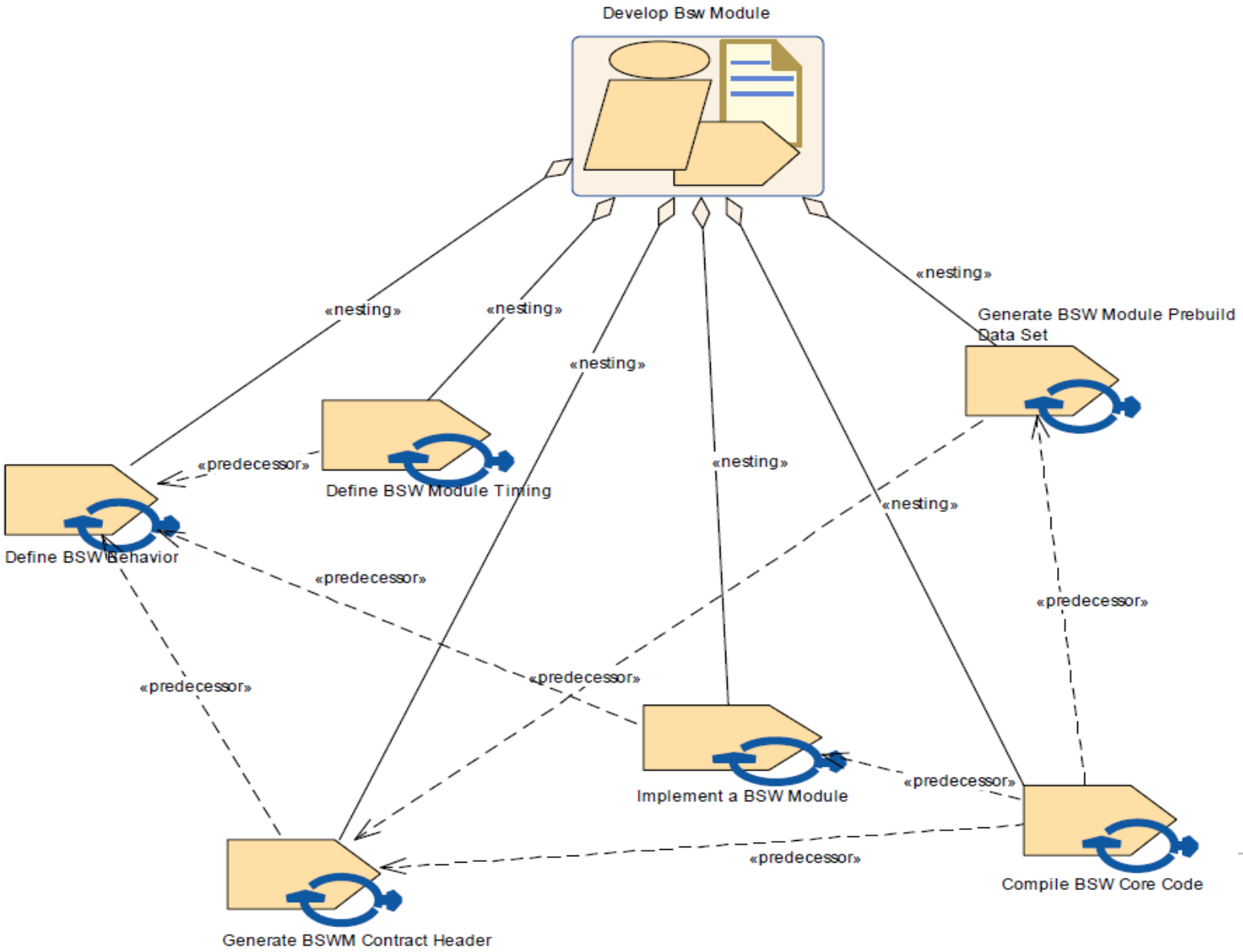
Relation Type	lement	Mul.
Predecessor	ommuni- eds	1
NestedBreakdownEle	ommuni- rix	1
NestedBreakdownEle	or Adjust at Map	1
NestedBreakdownEle	Software nt Imple-	1
NestedBreakdownEle	n Root	1
NestedBreakdownEle		
NestedBreakdownEle		
NestedBreakdownEle		
NestedBreakdownEle		



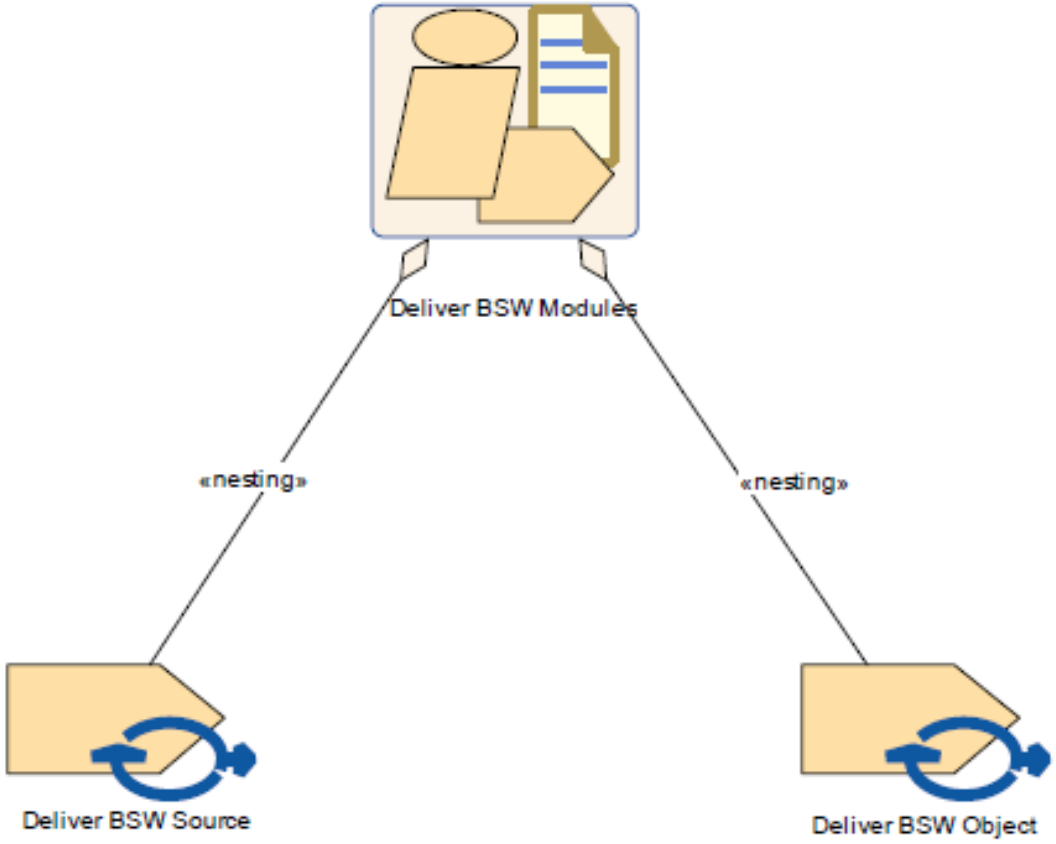
5.2.6 Develop BSW



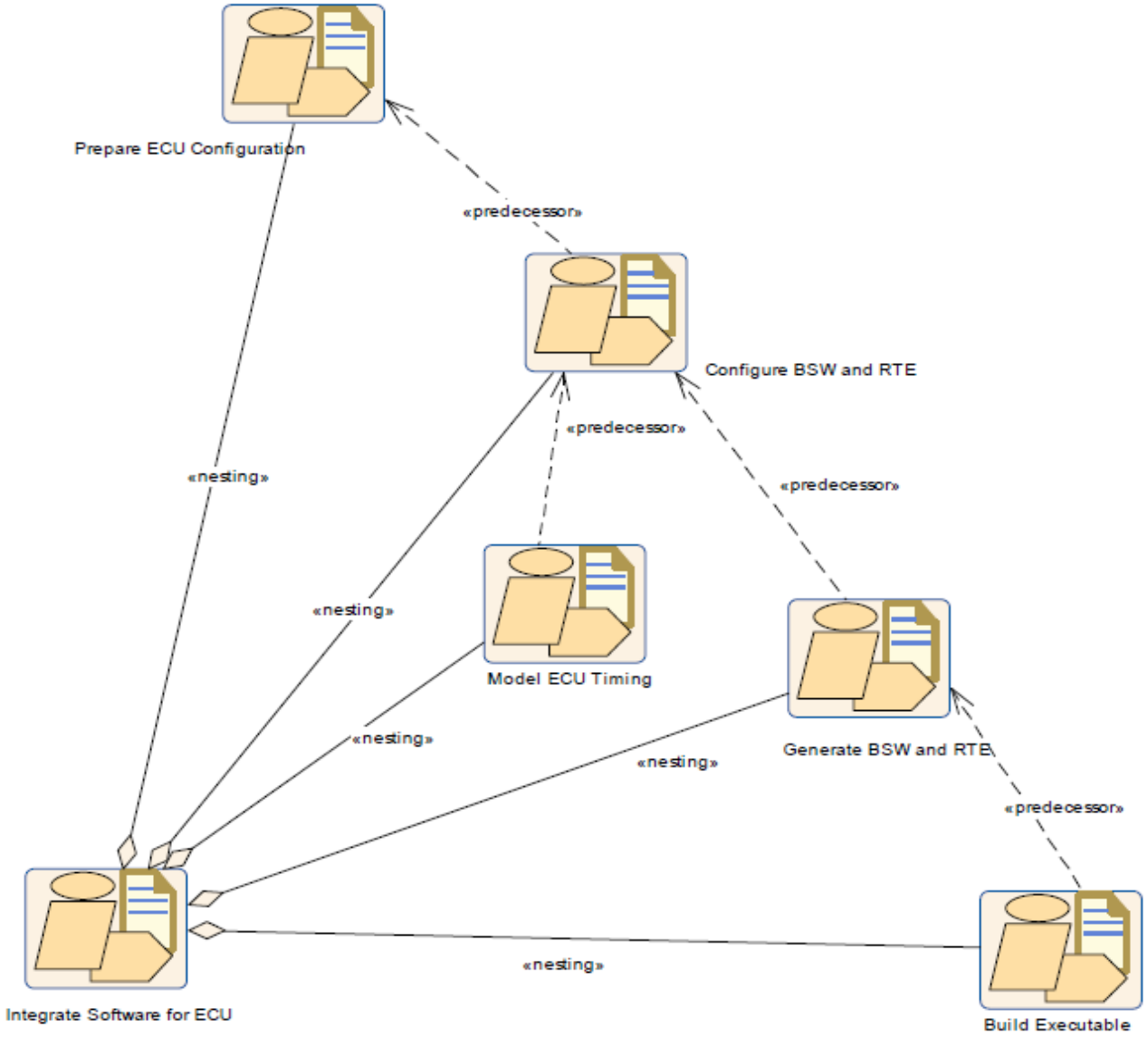
5.2.6 Develop BSW



5.2.6 Develop BSW



5.2.7 Integrate Software for ECU



Q&A