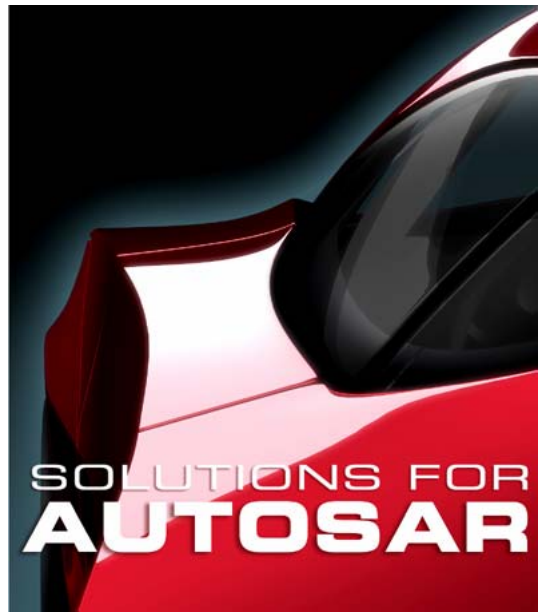


AUTOSAR

基础知识

vector

大纲

- 综述和目标

AUTOSAR入门 (introduction)

AUTOSAR方法论 (methodology)

AUTOSAR实时环境 (RTE)

AUTOSAR基础软件 (BSW)

Vector AUTOSAR实现

从CANbedded到AUTOSAR

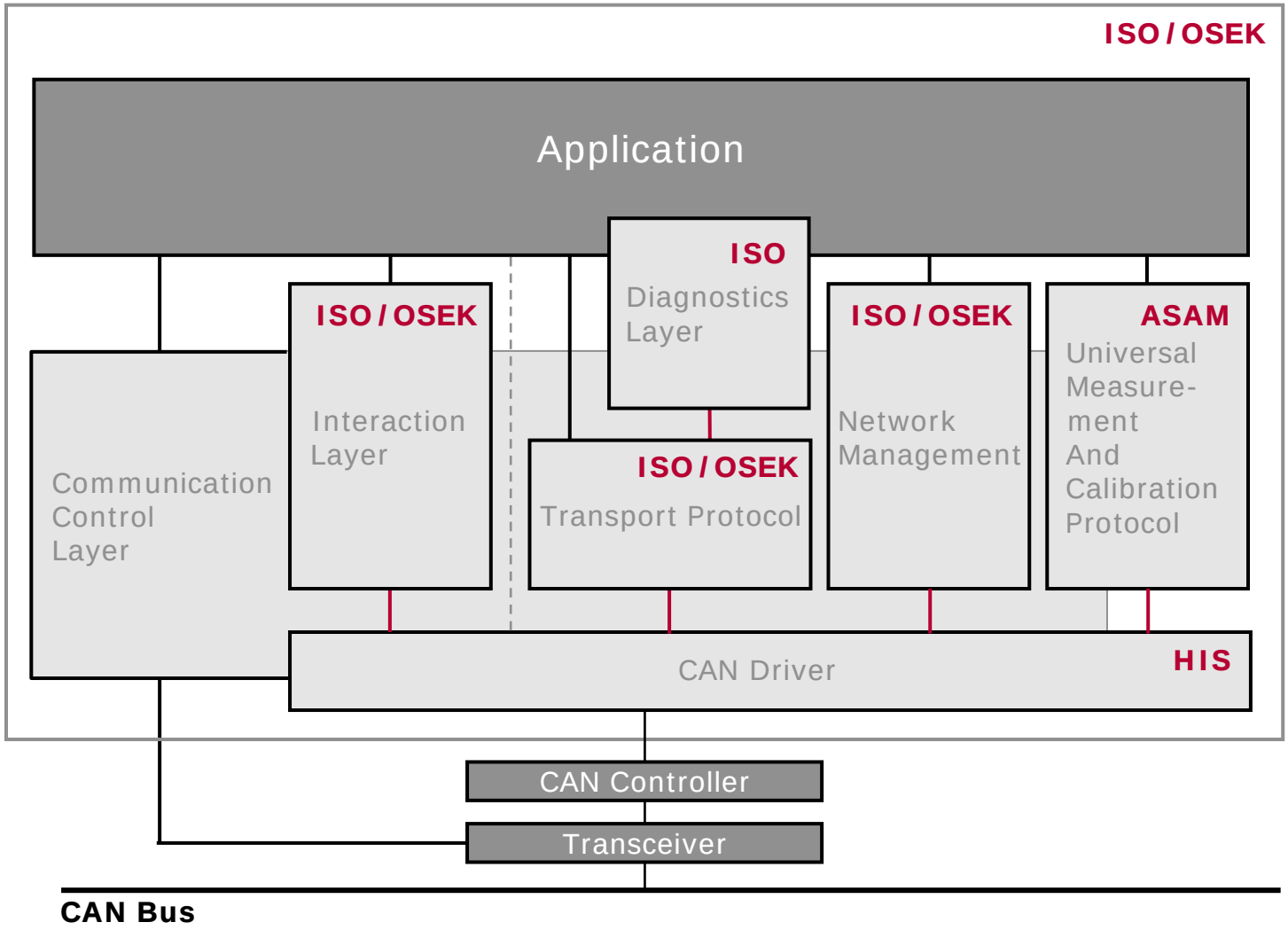
综述和目标

行业现状

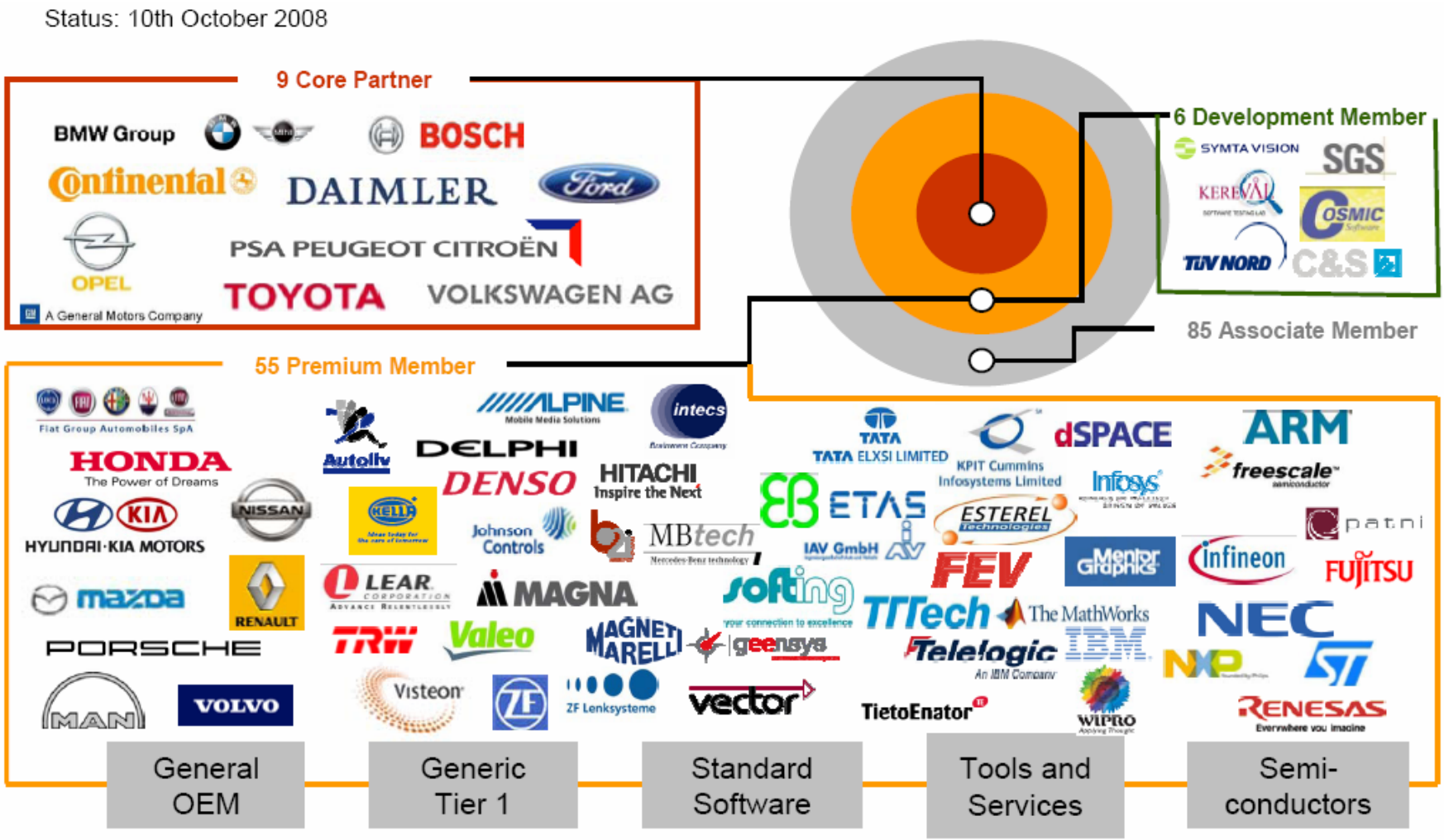
- 电子系统的**复杂性**不断增长
- **软件代码**量急速上升
- **生命周期差别**：整车的生命周期往往长于ECU的生命周期
- 嵌入式系统不支持**硬件抽象**
- 有限的软件**模块化**
- **重用性**差：当硬件（处理器型号）更换后，软件往往要推倒重写
- 五花八门的**硬件平台**

综述和目标

Vector此前的解决方案



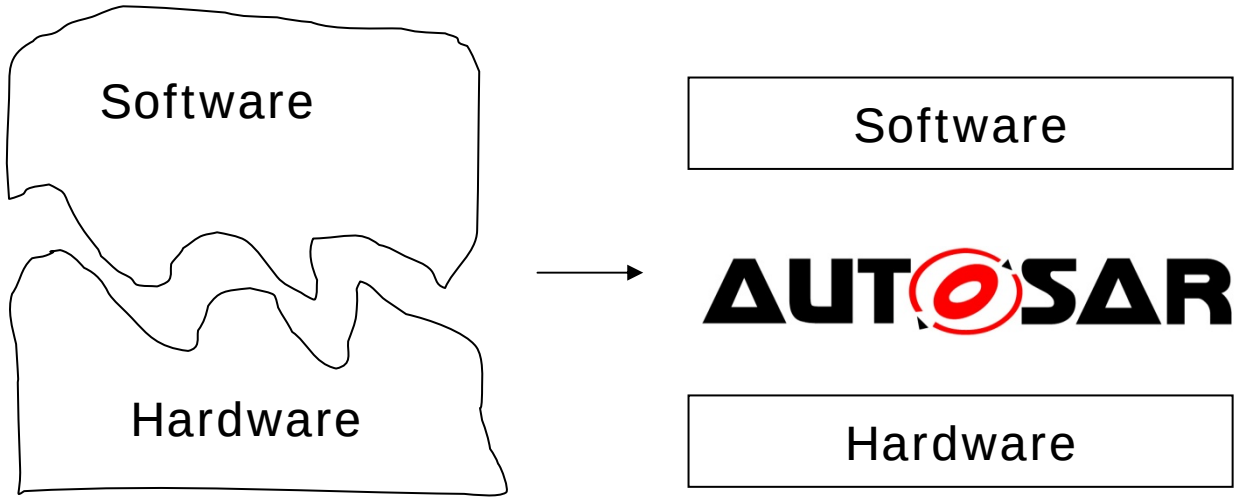
AUTOSAR成员



AUTOSAR 口号

Slogan:

“Cooperate on standards – compete on implementation”

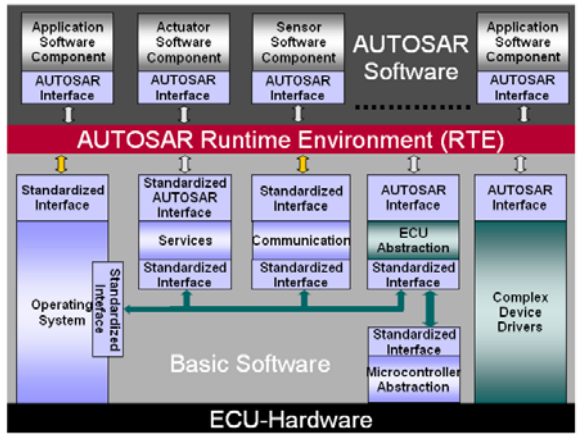


目标

AUTOSAR

AUTOmotive **O**pen **S**ystem **AR**chitecture

- 标准化
  - 软件接口
  - 交换格式
  - 方法论



- 将汽车系统的基础软件标准化为一个跨OEM的**“标准栈”**
- **集成**不同供应商生产的功能模块
- **适用**于不同的车辆及不同的车型

vector

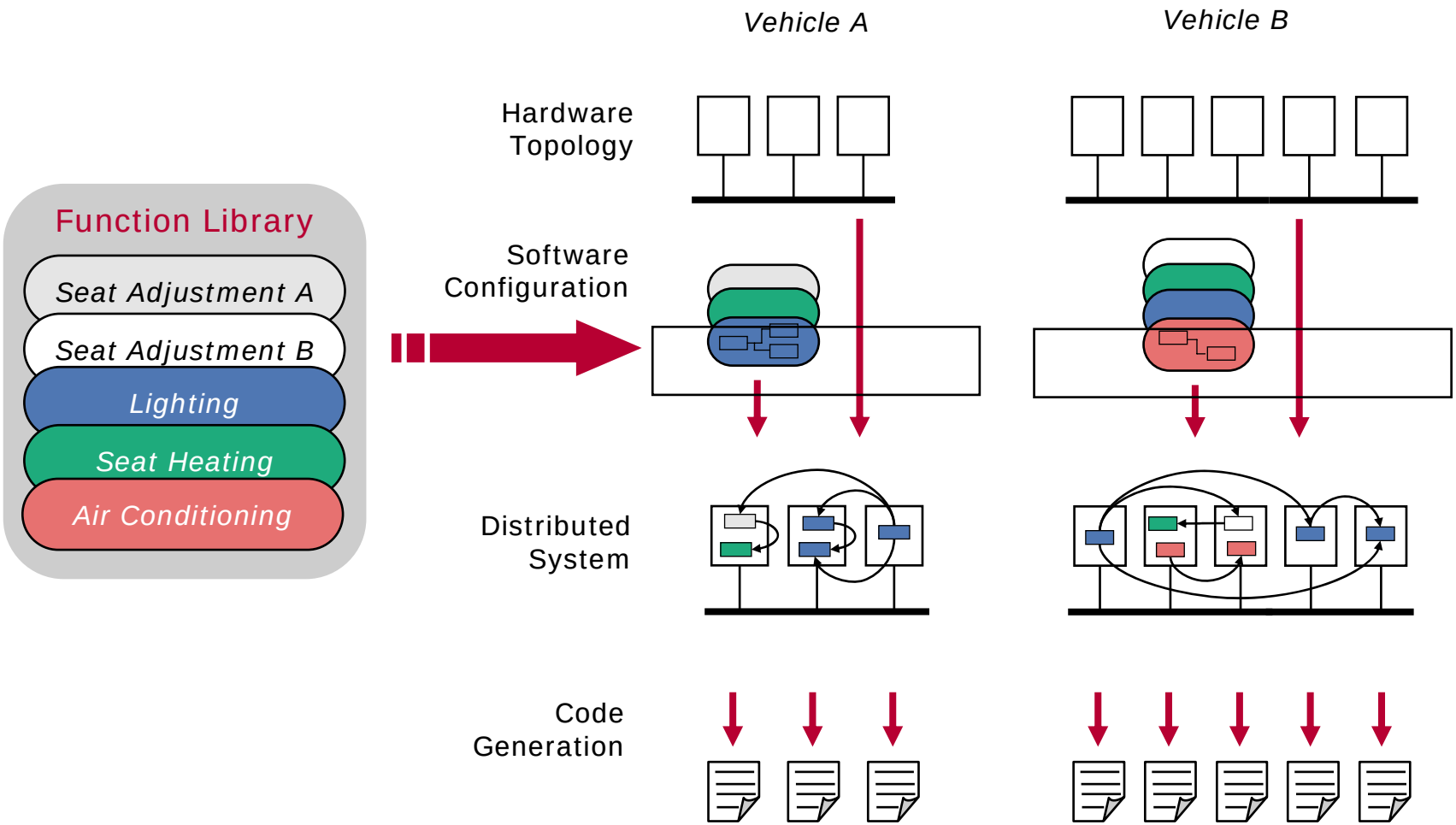
综述和目标

目标

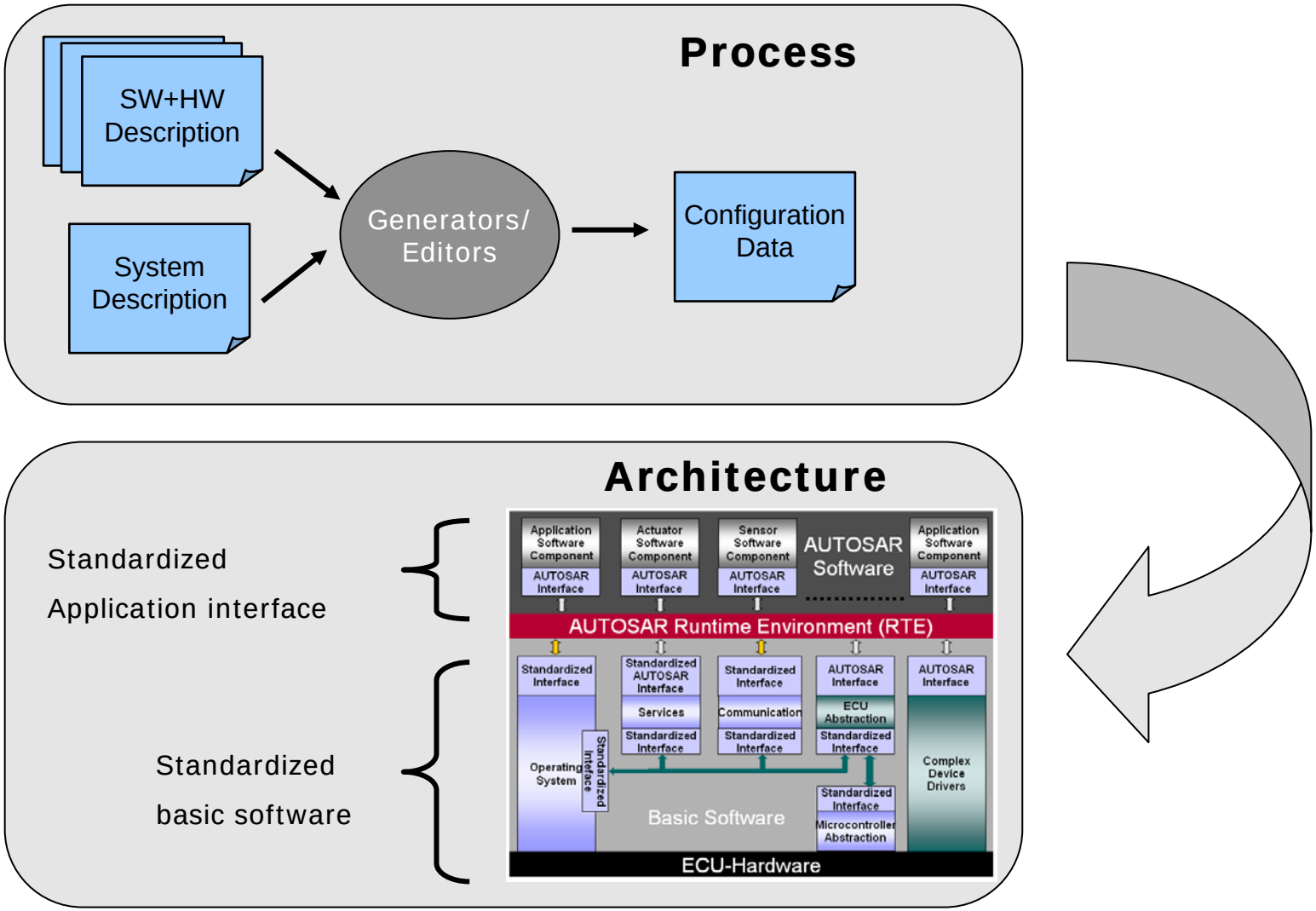
- **适用于**整个产品生命周期
- 从软件中把硬件**抽象**出来，对于不同硬件平台具有更大的灵活性
- 更多的**配置**而非实现
- 标准化AUTOSAR的**代码配置 / 建模**工具
- 通过对BSW的标准化提高了代码质量
- 竞争力只体现于对OEM的特殊功能要求的实现
- 在整个汽车生命周期中，软件可以不断更新或升级
- 重用性可以覆盖整个网络节点，甚至跨不同OEM

AUTOSAR 目标

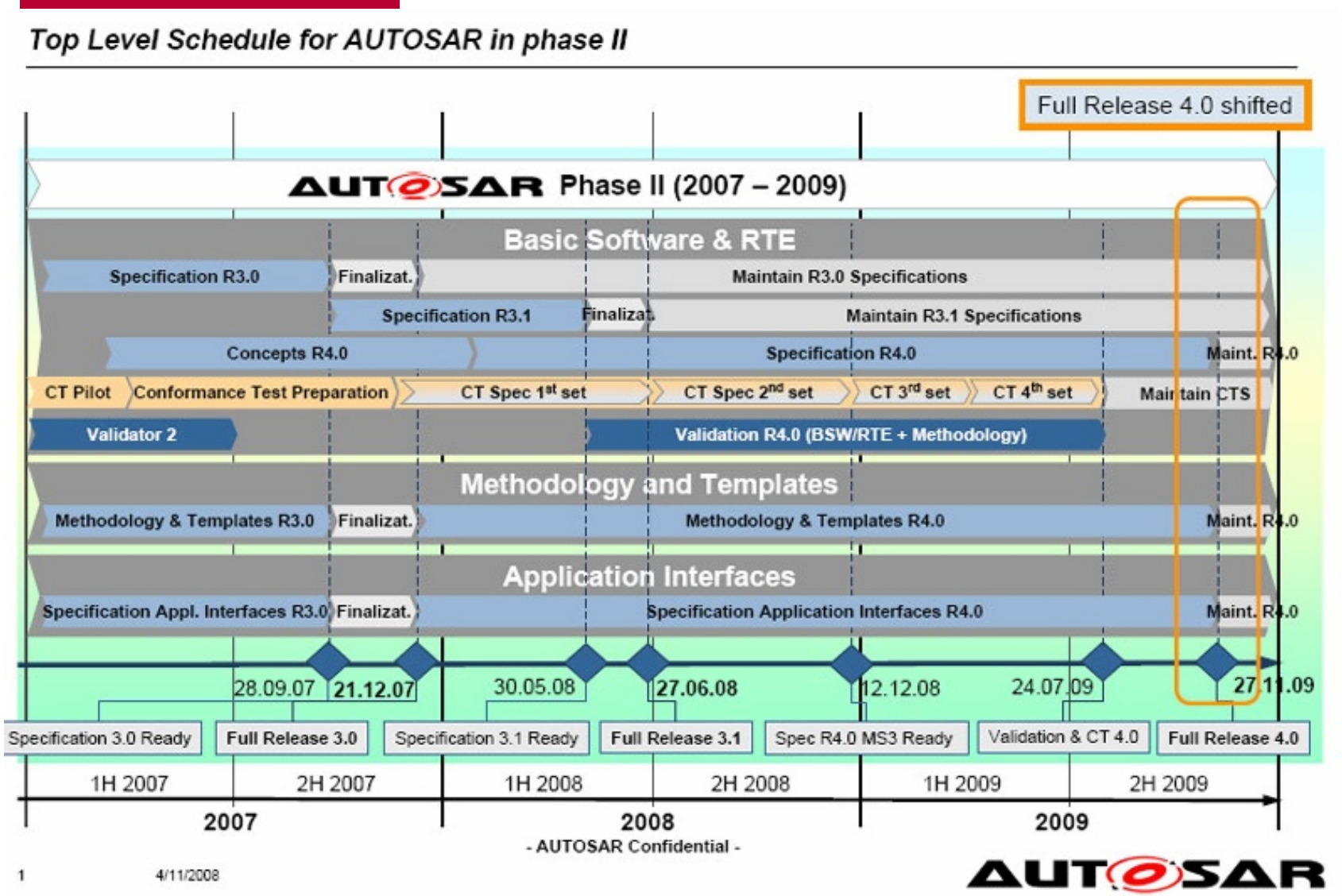
- 软件功能模块在不同车型之间被重用



AUTOSAR 解决方案



AUTOSAR 项目阶段



vector

大纲

综述和目标

> **AUTOSAR入门 (introduction)**

AUTOSAR方法论 (methodology)

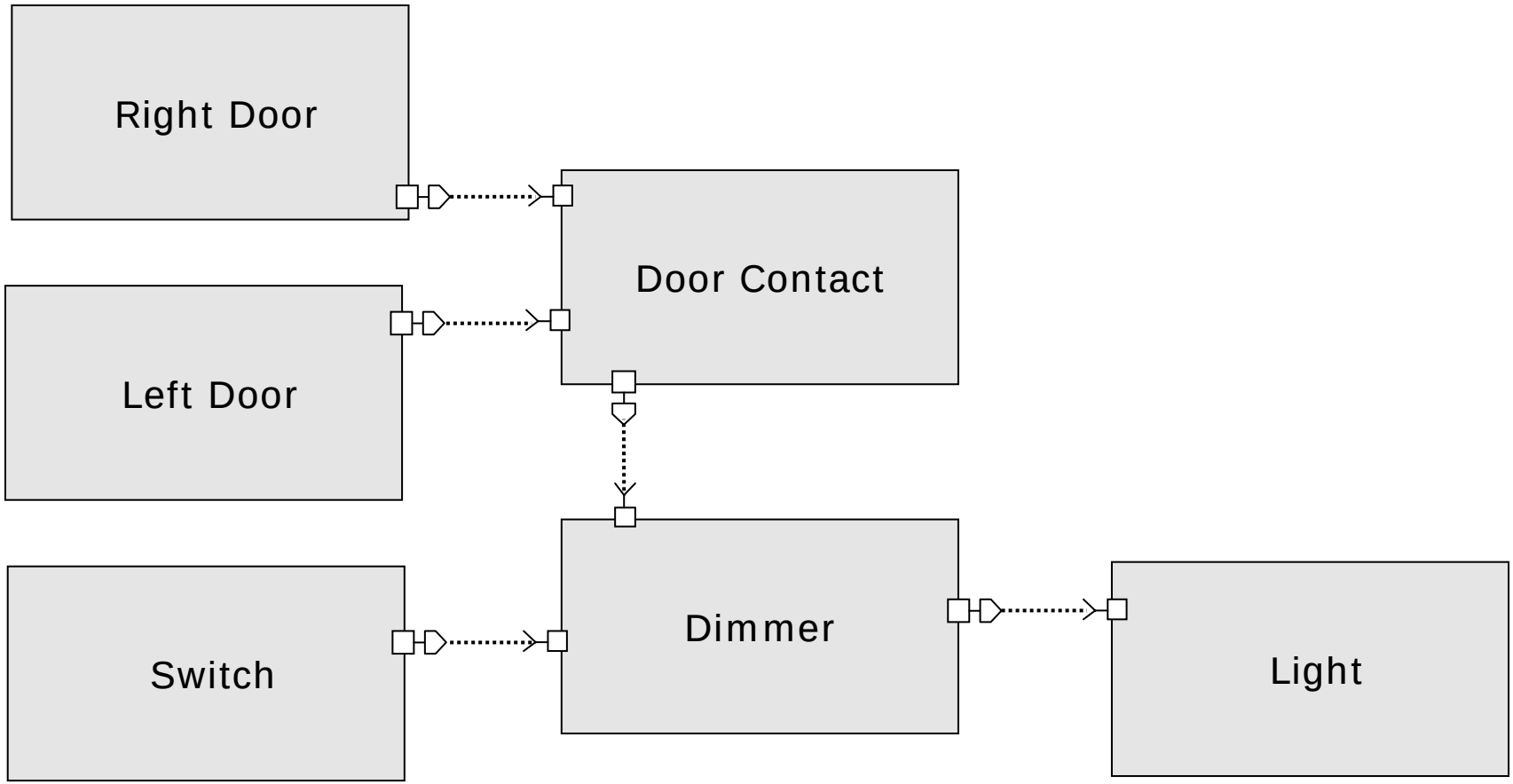
AUTOSAR实时环境 (RTE)

AUTOSAR基础软件 (BSW)

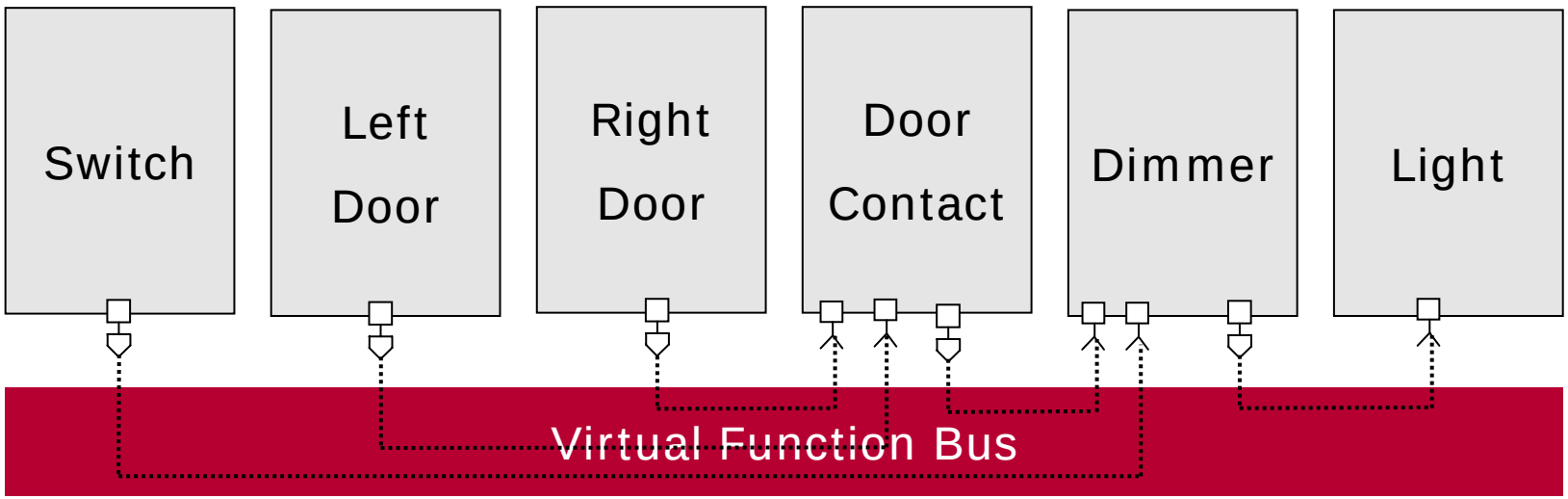
Vector AUTOSAR实现

从CANbedded到AUTOSAR

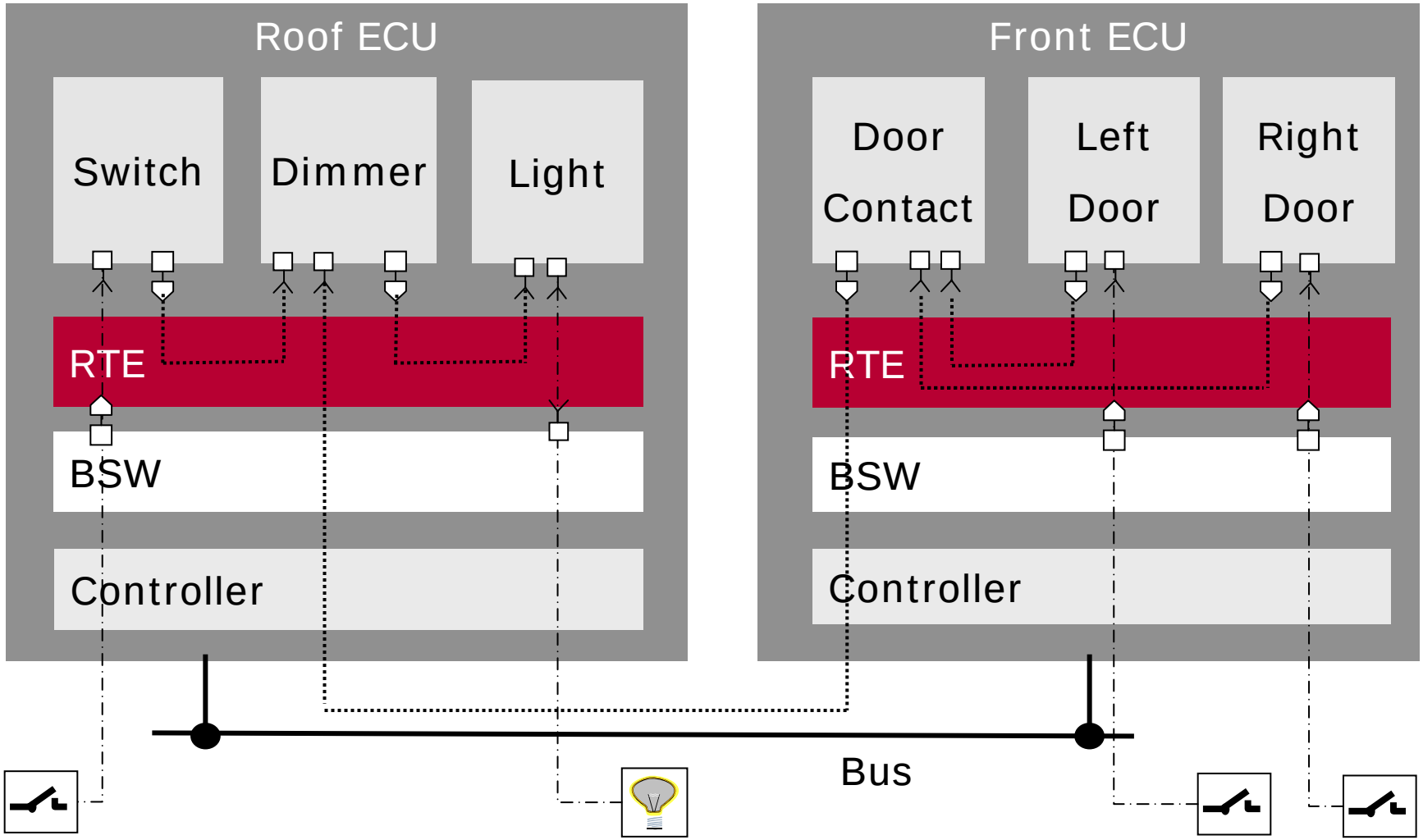
车灯控制系统的软件功能组件示意图



软件功能组件通信

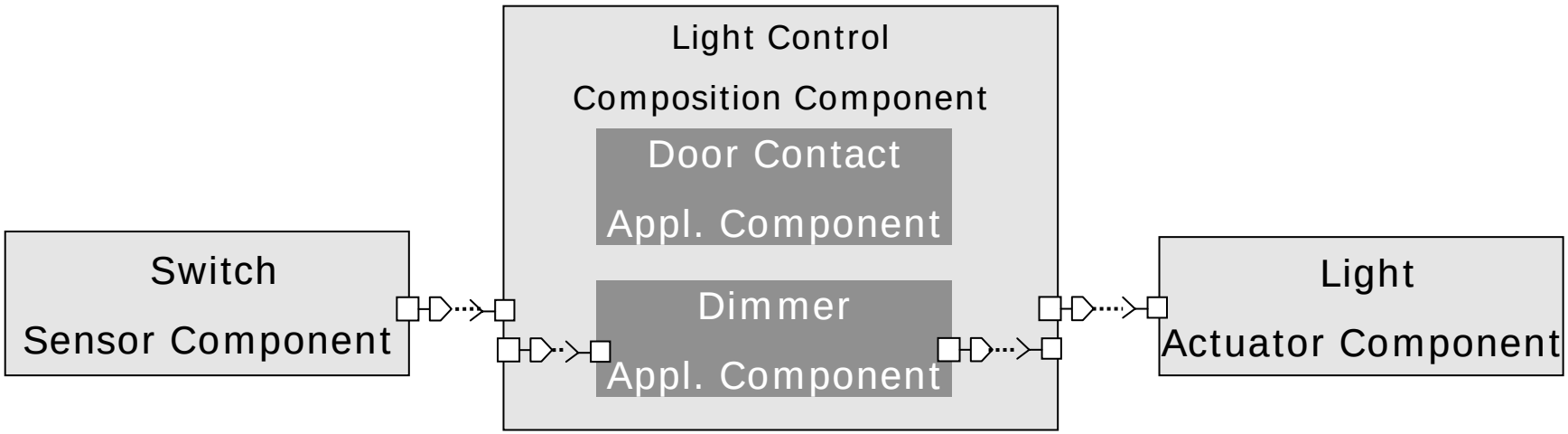


分布式软件组件



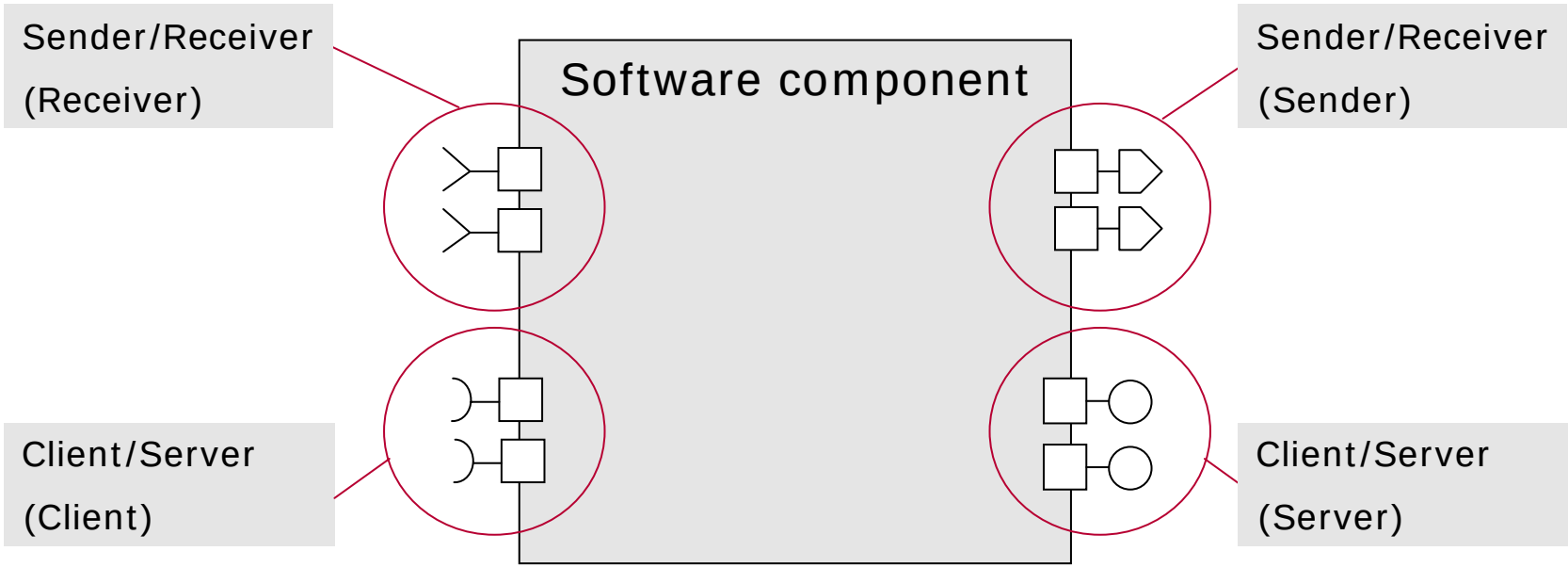
软件组件(SWC)的分类

- Atomic component (最小的逻辑单元，无法再分)
  - Application
    - 实现算法
  - Sensor/actuator
    - 为Application提供I/O量
    - 与ECU绑定 (不像Application那样能在各ECU上自由映射)
- Composition -> 数个SWC的逻辑集合

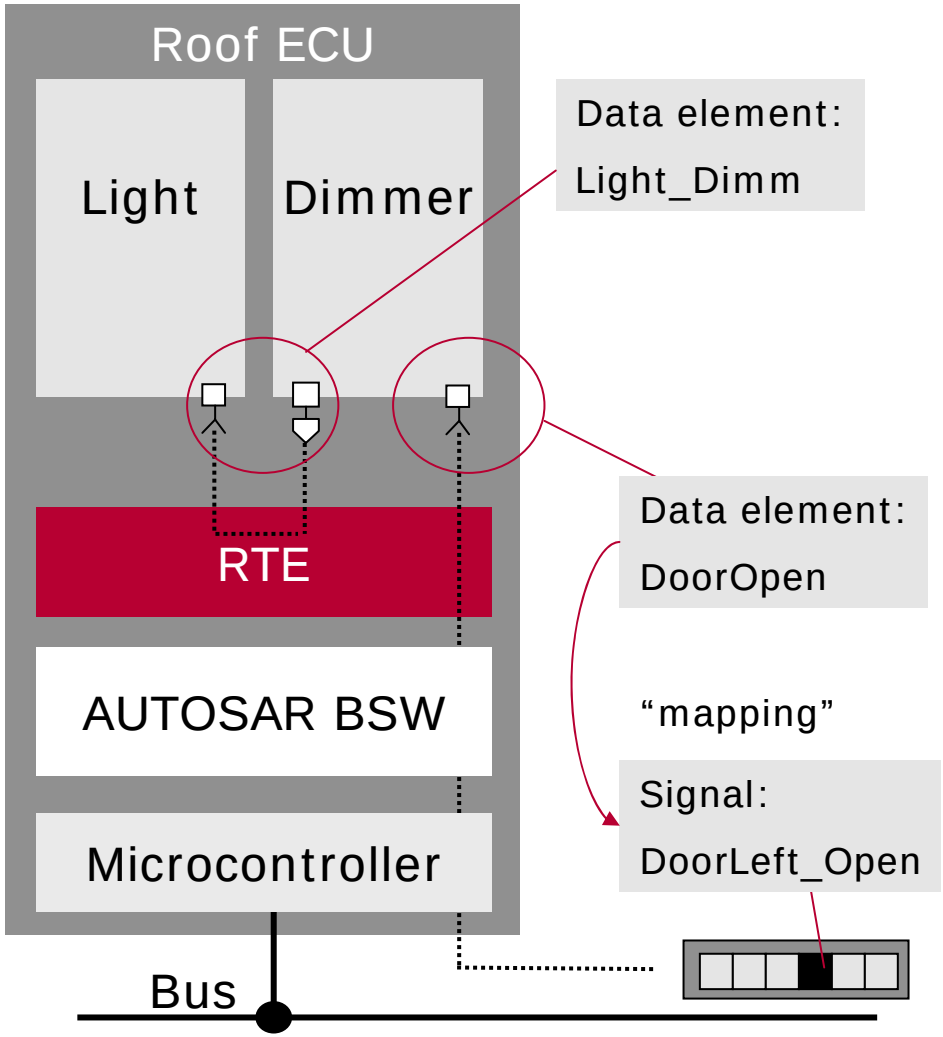


端口(Ports)

- SWC 的组成之一:
  - Ports
    - 和其他SWC的通信端口
    - 通信内容: Data elements (S/R) 与 operations (C/S)



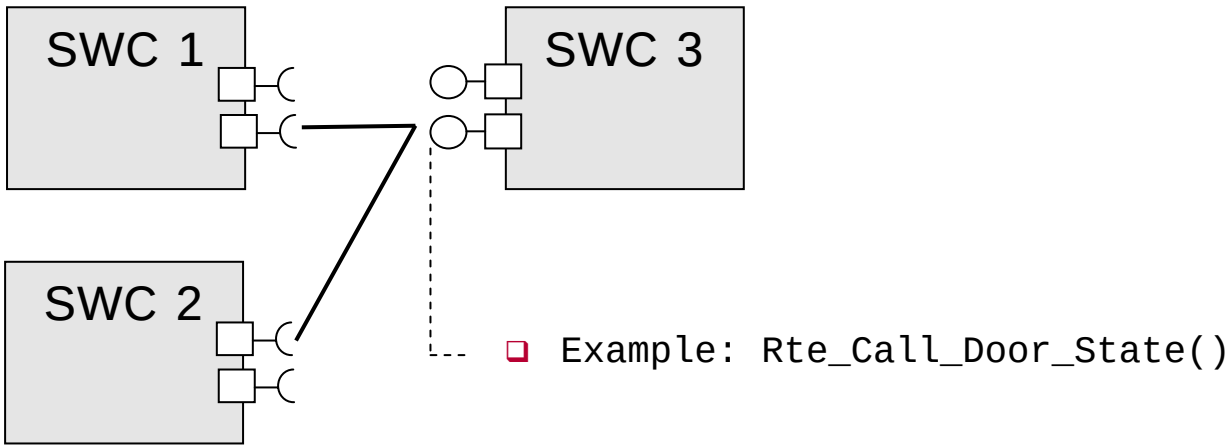
发送-接收端口(Sender-Receiver)



- 传输数据
- 一个port可以包含多种data element
- 如果一个data element要通过总线传输， 那么它必须与一个signal对应起来
- DE既可以是简单的数据类型(integer, float), 也可以是复杂类型(array, record)
- 通信方式: 1:n or n:1

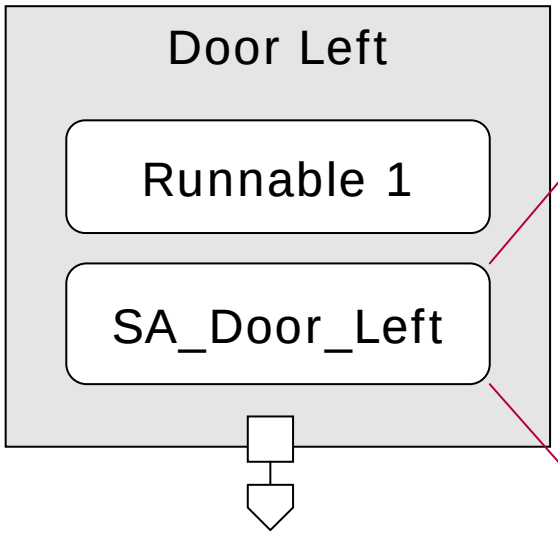
客户端-服务器端口(Client-Server)

- 提供Operation服务
- 通信方式: 1:1 or n:1 （与S/R对应）
- 同步或异步
- 一个C/S port包含多种operations
- Operations可以被单个调用



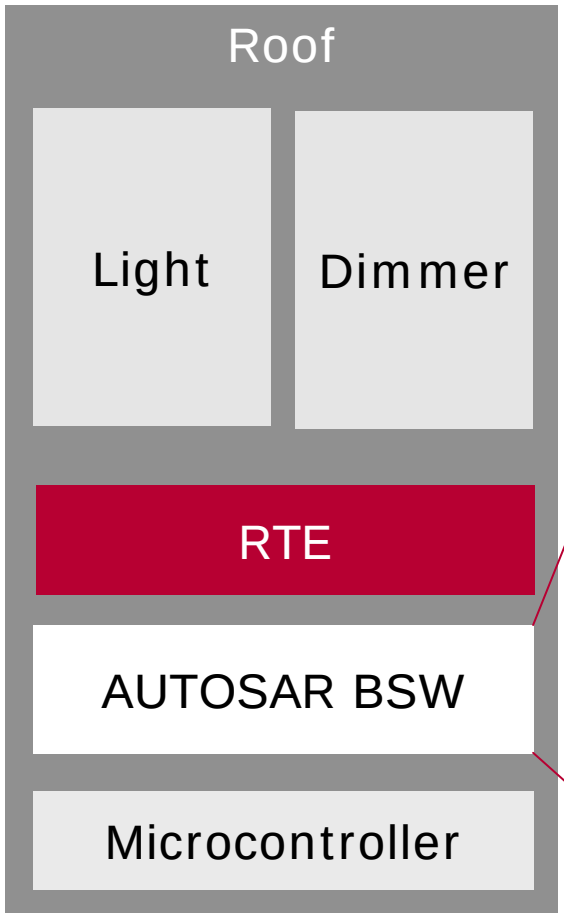
可运行实体(Runnables)

- SWC 的组成之二:
  - Runnable entities (简称Runnables)
    - 包含实际实现的函数（具体的逻辑算法或者操作）
    - Runables由RTE周期性、或事件触发调用
      (如，当接收到数据)

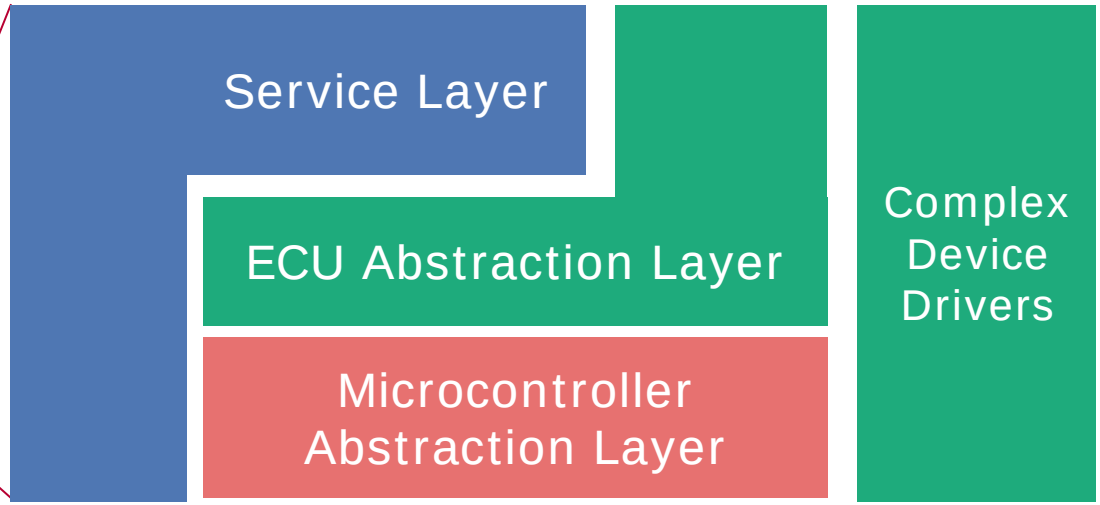



```
// triggered every 500 msec
void SA_Door_Left()
{
  Std_ReturnType status;
  boolean DoorOpen;
  ...
  status=Rte_Write_<Port>_<Data>(DoorOpen);
}
```

架构体系

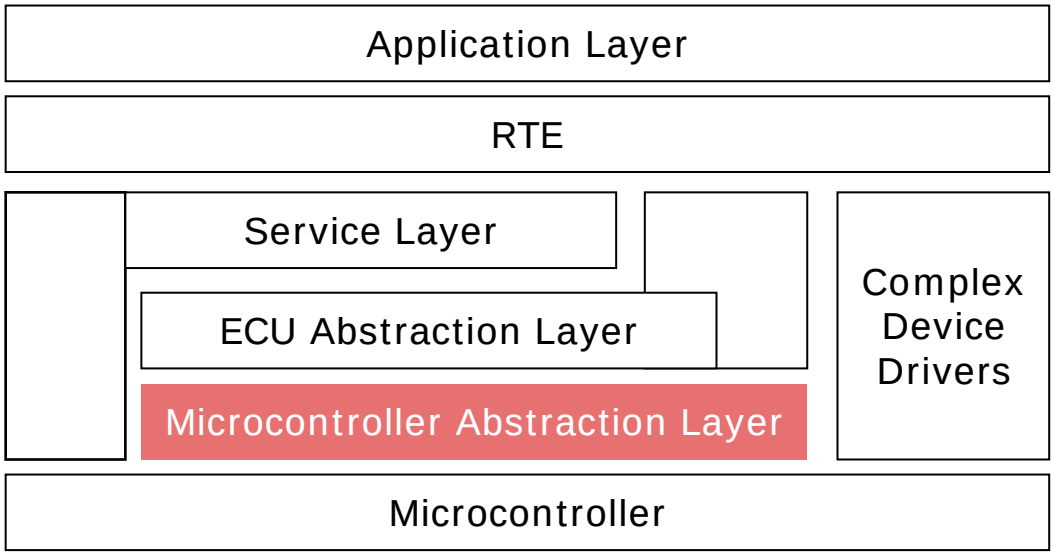


- 80个BSW被抽象划分为3个层面



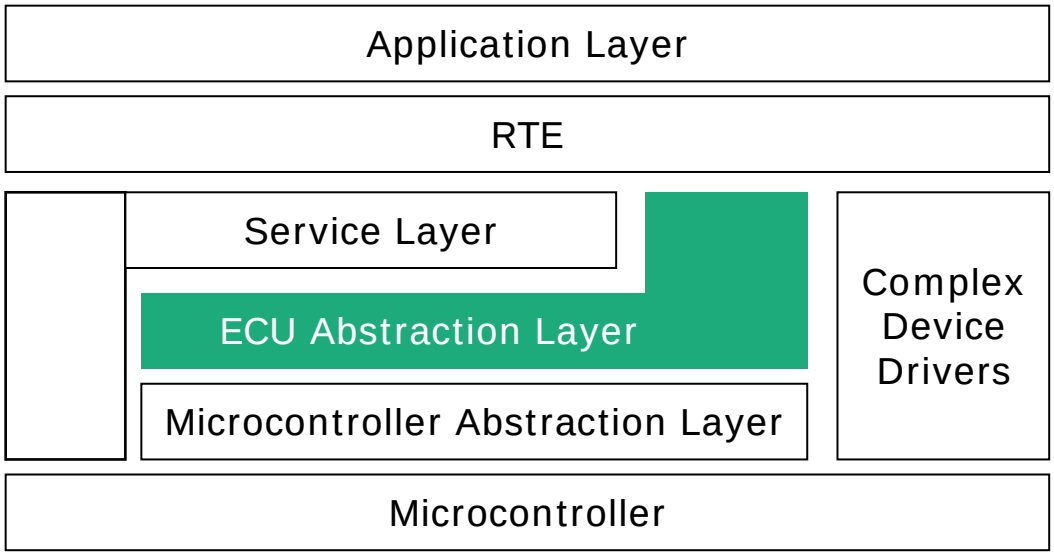
微控制器抽象层(Microcontroller Abstraction Layer)

- 目的:
  - 使上层软件与微处理器型号无关
- 功能:
  - 包含MCU中内部外设的驱动
  - 包含使用MCU内存映射的外部设备的驱动



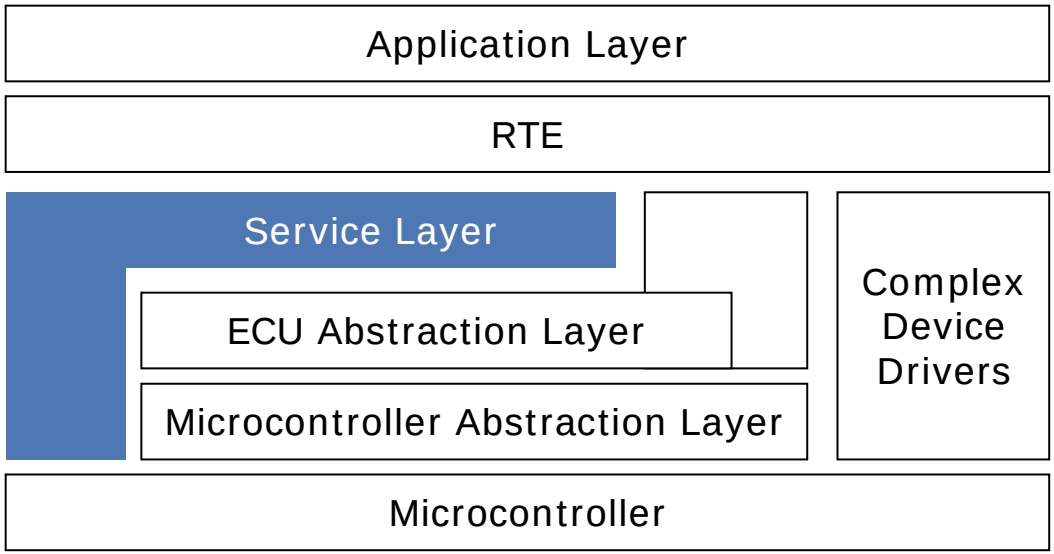
ECU抽象层(ECU Abstraction Layer)

- 目的:
  - 使上层软件与ECU硬件设计无关
- 功能:
  - 包含ECU板上外部设备的驱动
  - 内部设备与外部设备的接口(I/O)



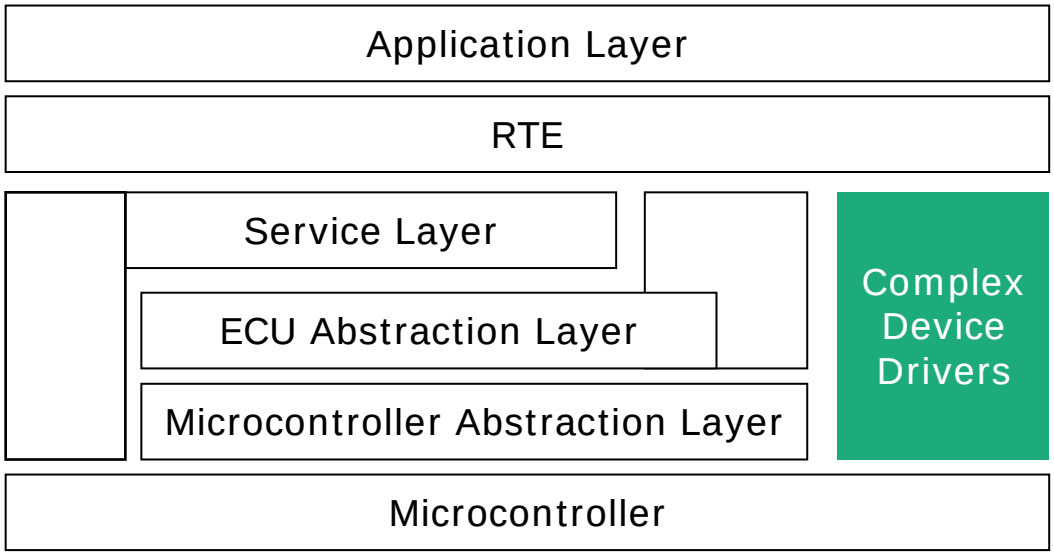
服务层(Service Layer)

- 目的:
  - 提供给应用程序可用的服务
- 功能:
  - 诊断, 非易失性内存管理, 操作系统, 通信
  - 内存和ECU管理



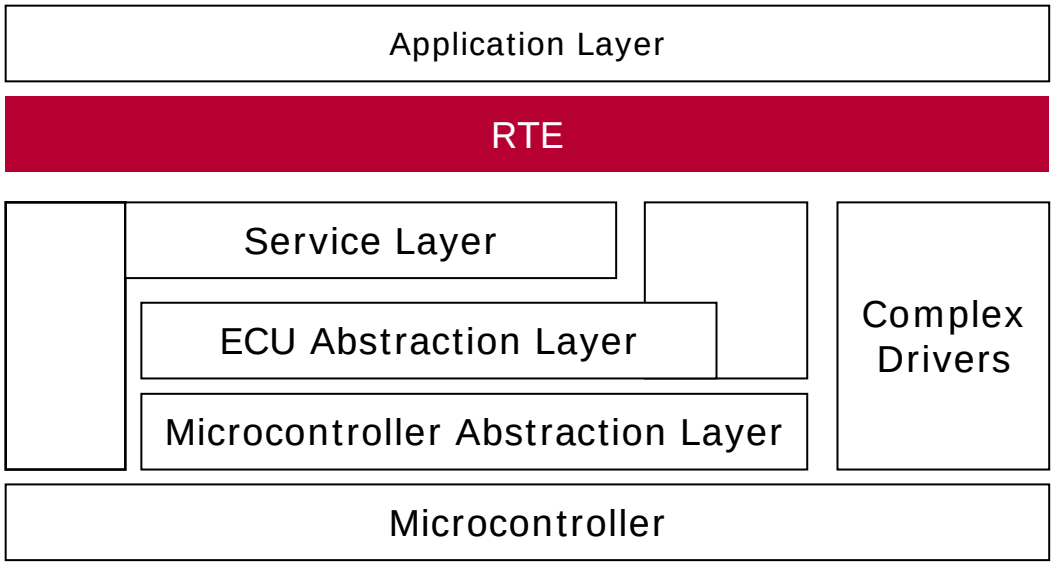
复杂设备驱动(Complex Device Drivers)

- 目的:
  - 提供复杂传感器和执行器的驱动
- 功能:
  - 重要的应用模块可以直接访问硬件资源
  - 例如: 喷油量控制, 胎压监测

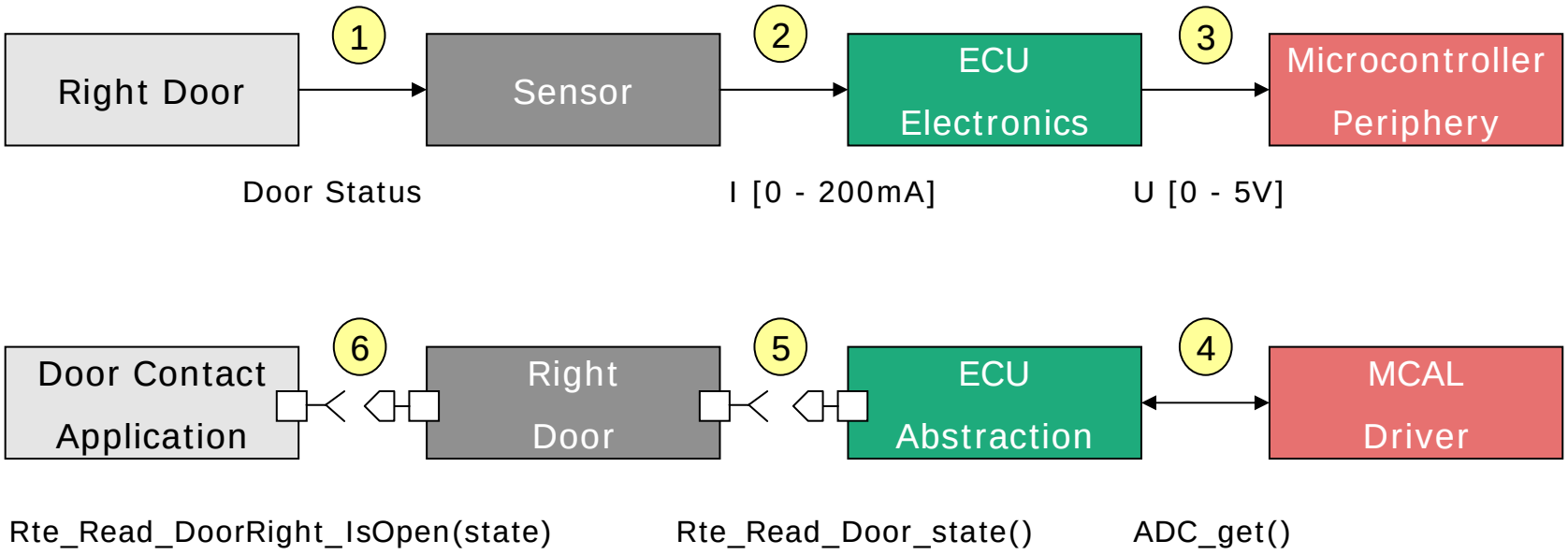


运行时环境(RTE)

- 目的:
  - 使SWC与ECU的映射无关
- 功能:
  - 提供通信服务的中间层 (ECU内部/间通信)



- 从传感器到应用程序的过程



GD35

Hier sieht man 2 Ebenen
a) Hardware Ebene mit Bauteilen und Elektronik
b) Software Ebene mit Treibern und Applikationssoftware
Elmar Gschwind, 2006-8-14

大纲

综述和目标

AUTOSAR入门 (introduction)

> **AUTOSAR方法论 (methodology)**

AUTOSAR实时环境 (RTE)

AUTOSAR基础软件 (BSW)

Vector AUTOSAR实现

从CANbedded到AUTOSAR

AUTOSAR方法论?

- 方法论就像菜谱，告诉我们做菜要用那些配料，需要哪些步骤
- 一本好菜谱包含：
  - Who, what, when, how and for which purpose

OEM – Supplier

Motivation

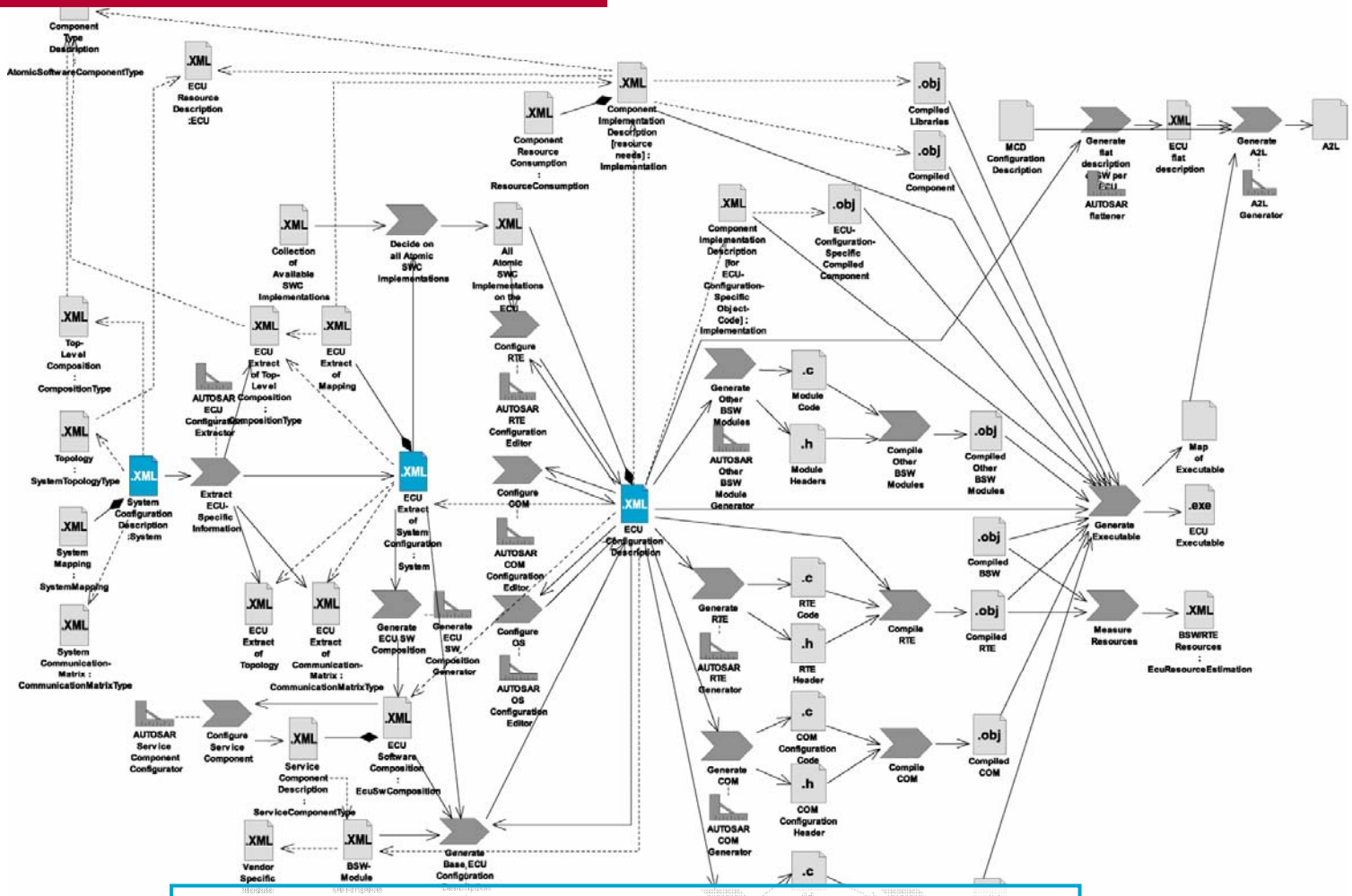
Tools

Data

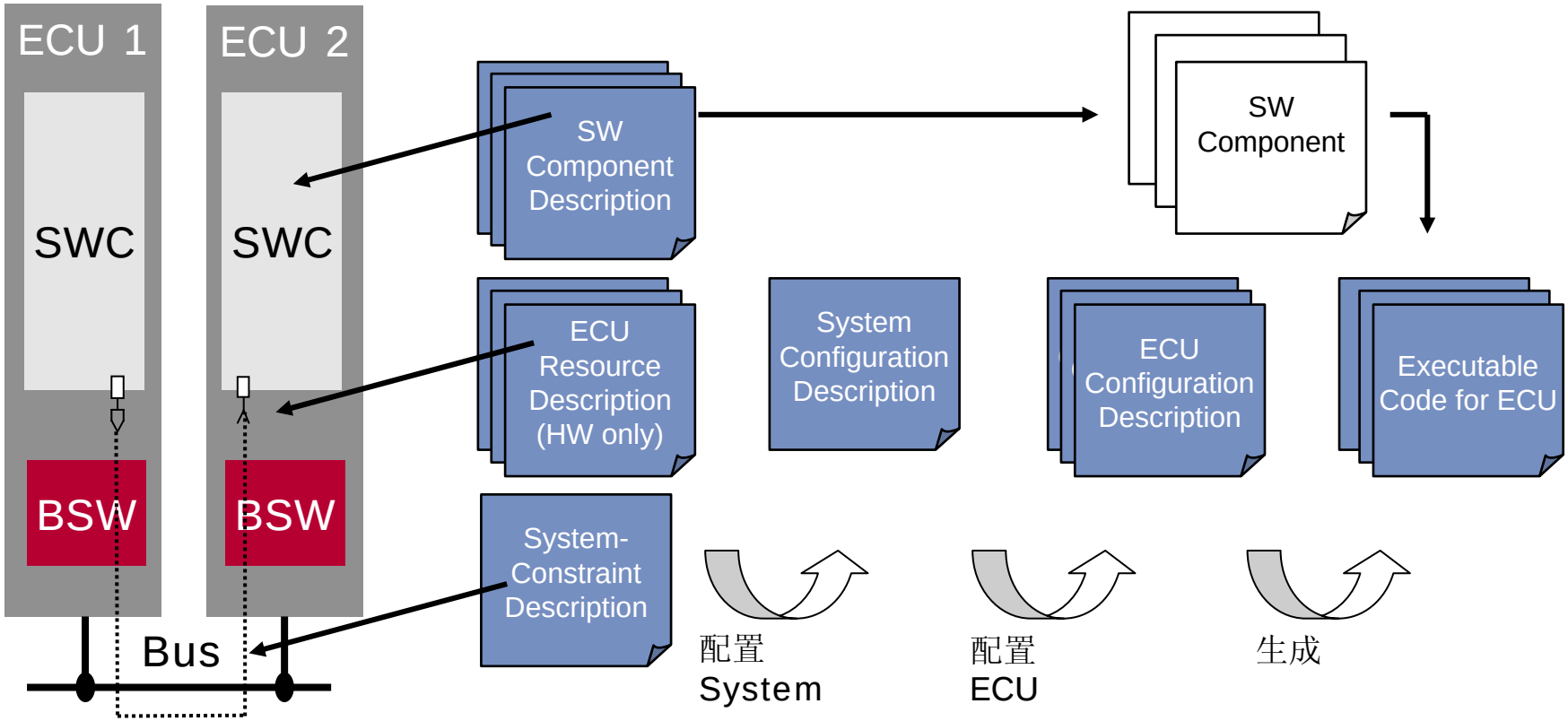
Integration

Migration

AUTOSAR Methodology全局观

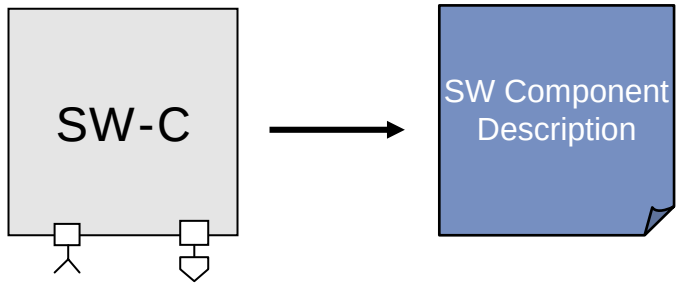


- System Configuration Description
- ECU Extract of System Configuration Description
- ECU Configuration Description



Software Component Description

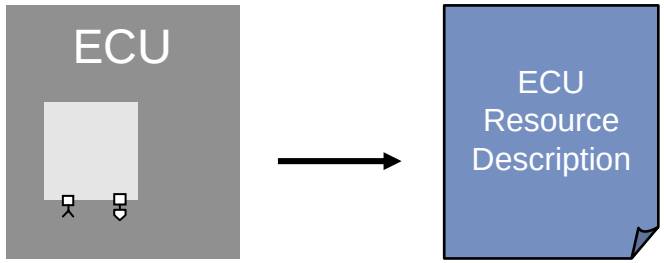
AUTOSAR 中“Software Component Description” 包含下列信息:



- 该SWC用到或被用到的Operation和Data
- SWC对基础构架(network)和对硬件(latency times, timing, etc.)的要求
- SWC使用的资源 (memory, CPU time, etc.)
- 运行机制(repetition rate)
- SWC软件接口

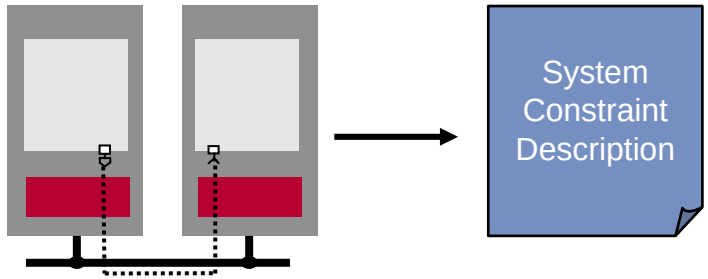
ECU Description

AUTOSAR中 “ECU Resource Description” 包含下列信息:



- 描述了使用到的硬件:
  - Sensor, actuator 传感器，执行器
  - Memory 存储器
  - Processor 处理器
  - Communications periphery 通信外部设备（如 收发器）
  - Pin assignments 引脚分配

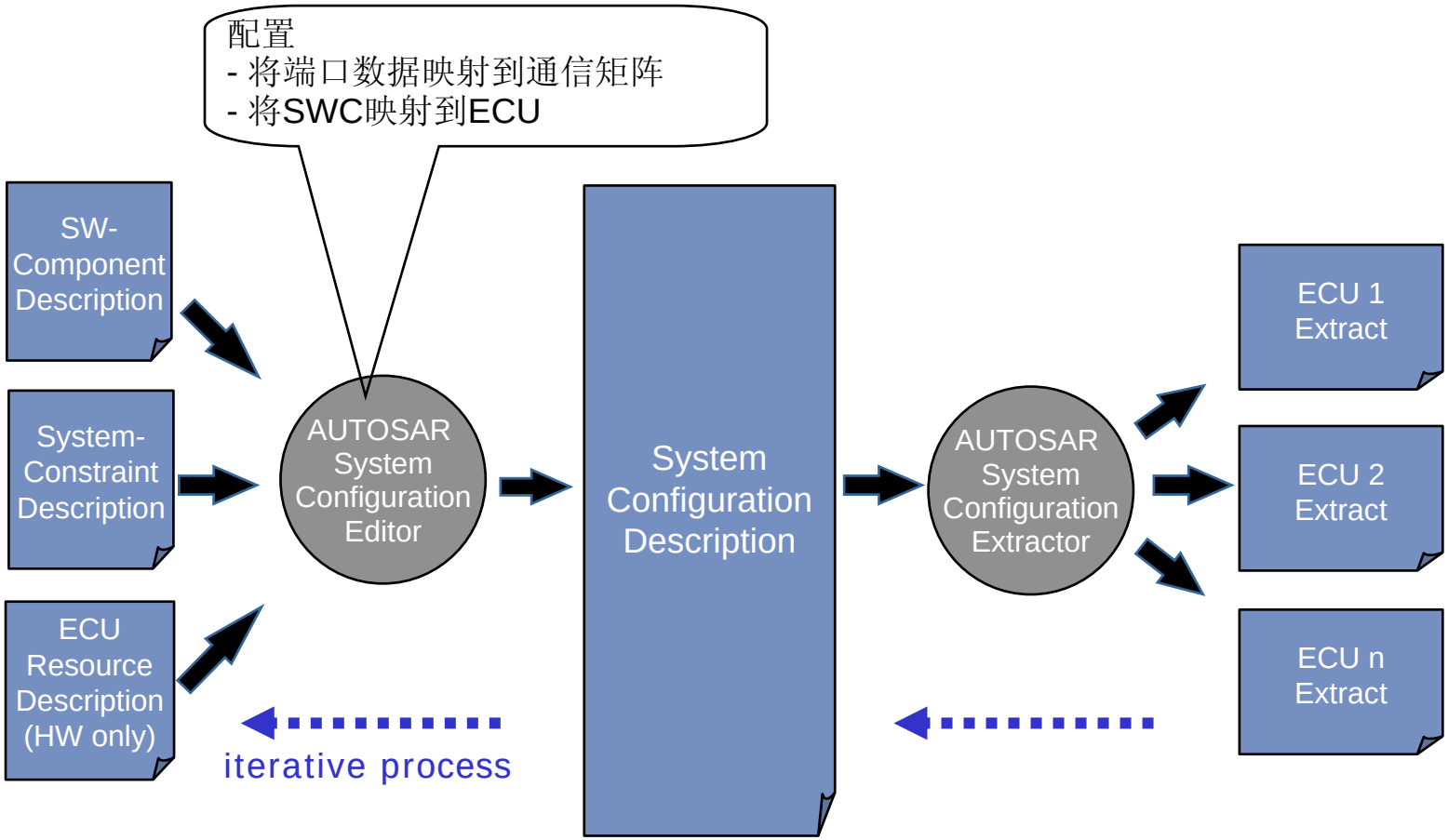
在AUTOSAR “System Constraint Description” 中包含下列信息:



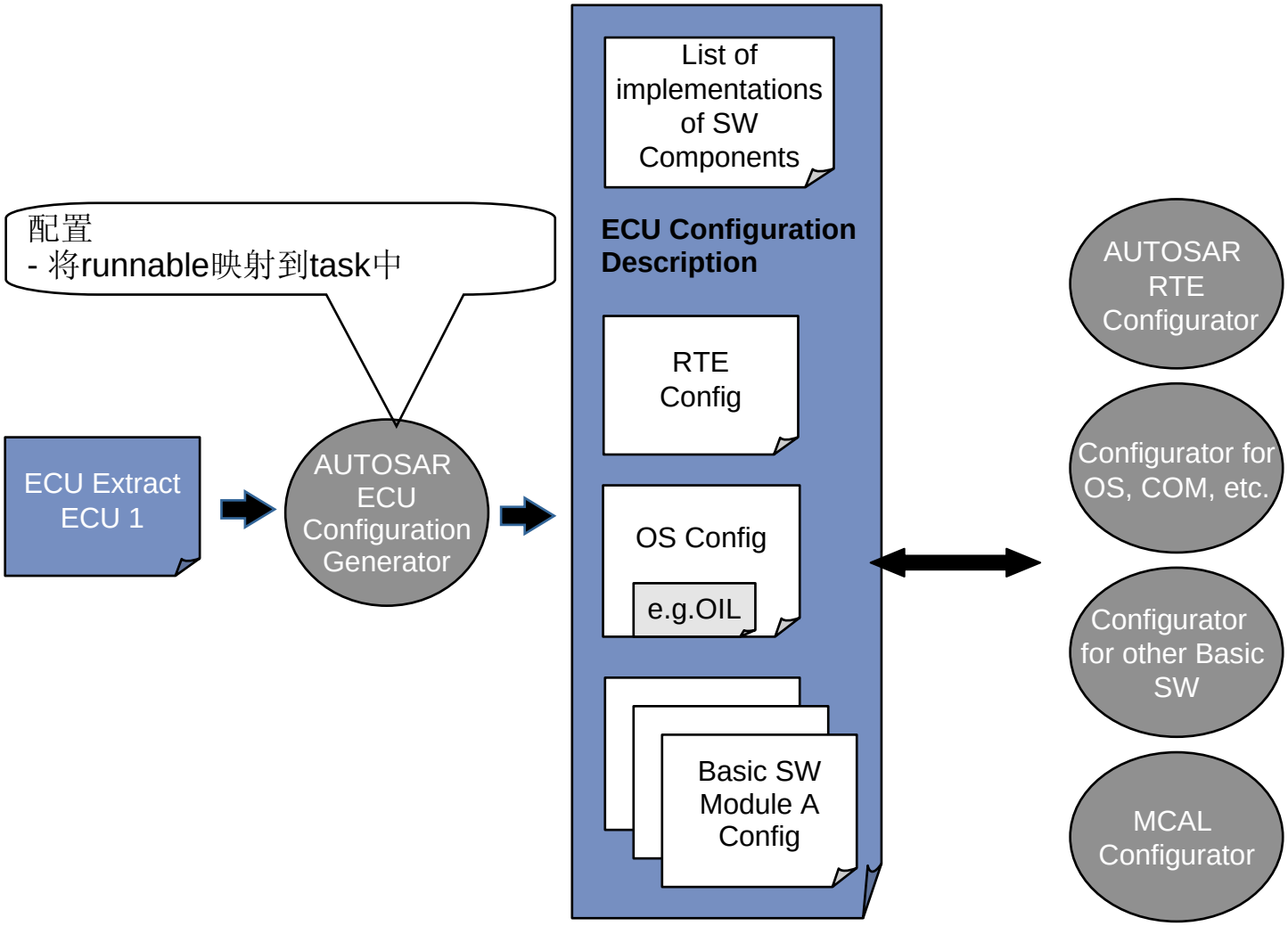
- Information on network topologies 网络拓扑
- Limitations (“Constraints”) 限制
- Protocol 协议
- K-matrix 通信矩阵
- Baud rate, timing 波特率，定时
- ECU映射

AUTOSAR 方法论

系统配置

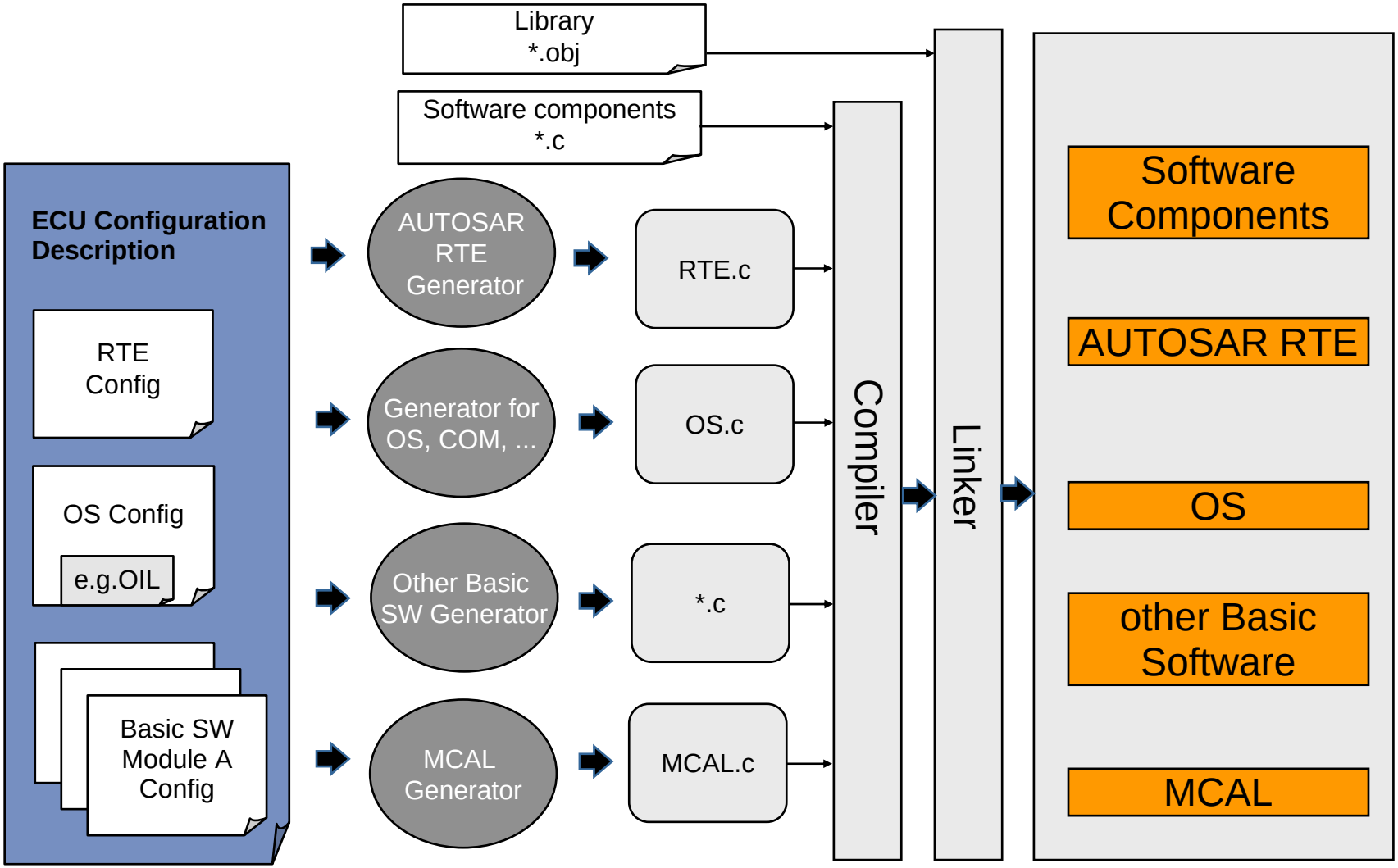


ECU 配置

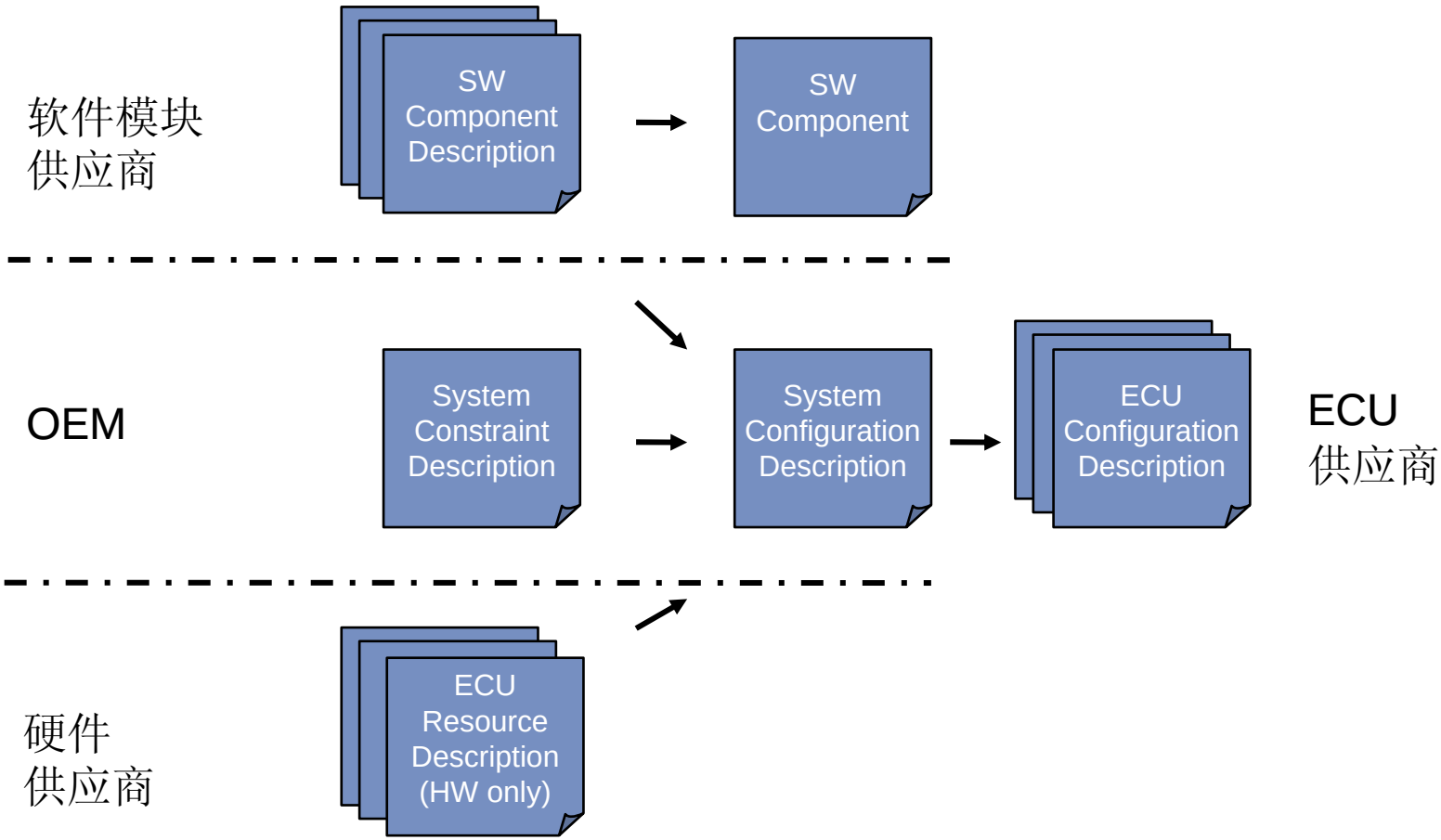


AUTOSAR 方法论

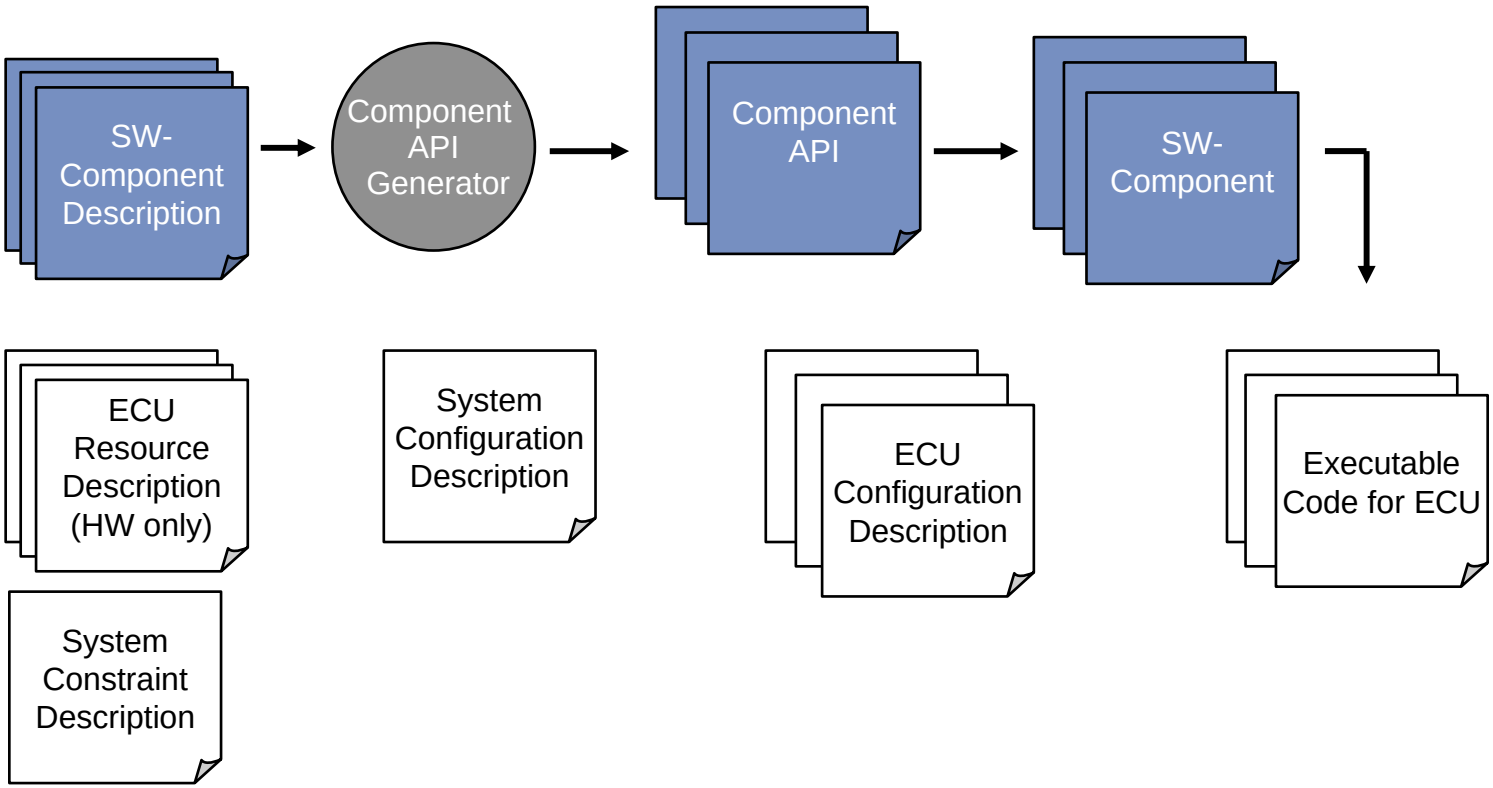
实现 ECU



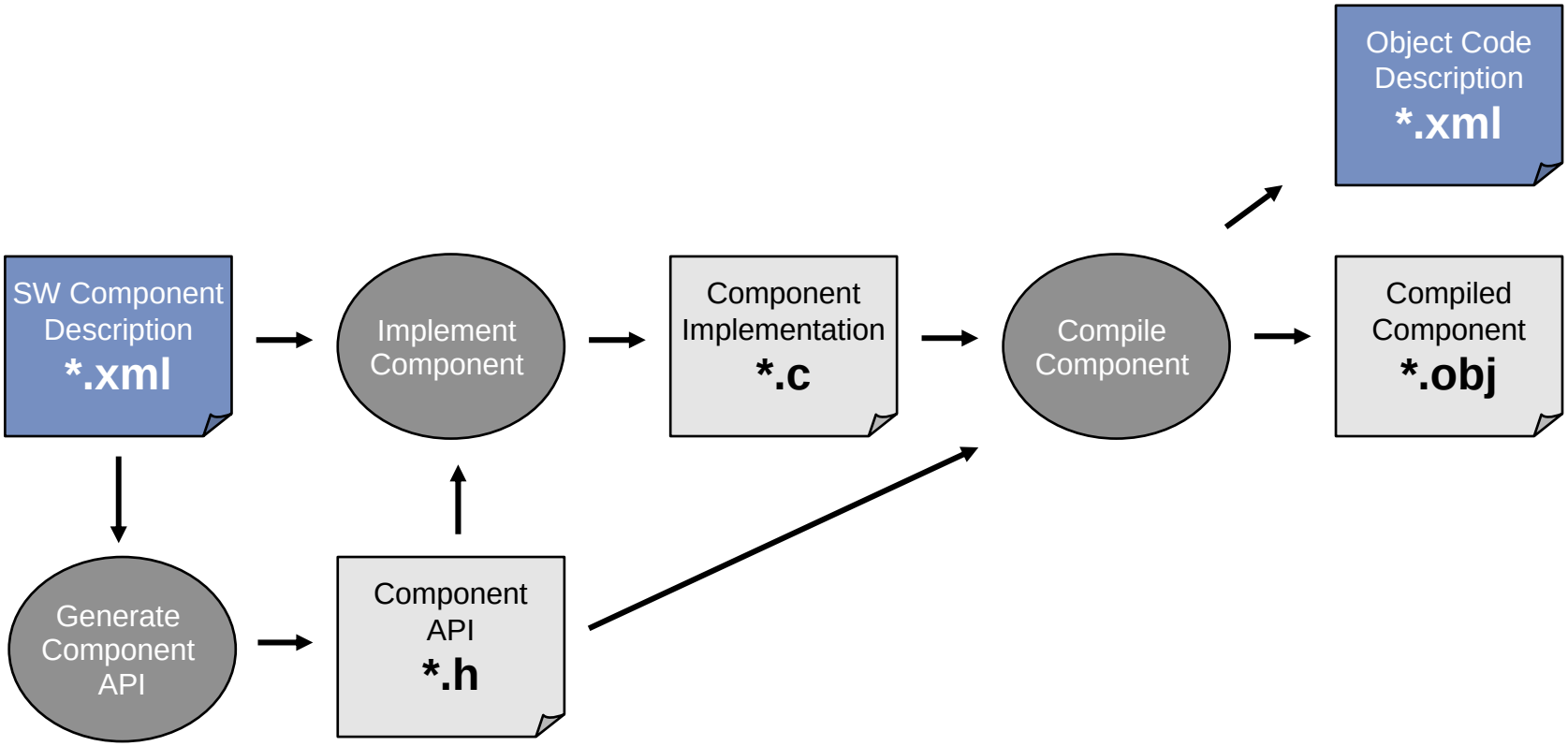
角色分工



SWC开发流程



SWC开发流程

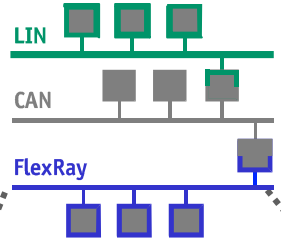


AUTOSAR 方法论

Vector开发工具

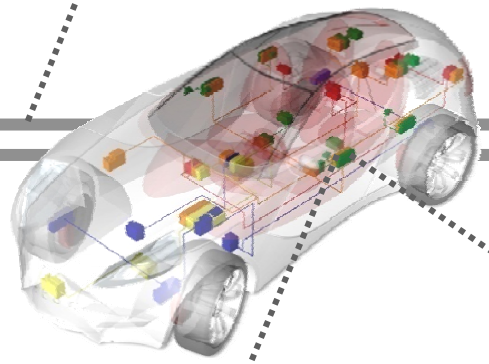
System Design

OEM



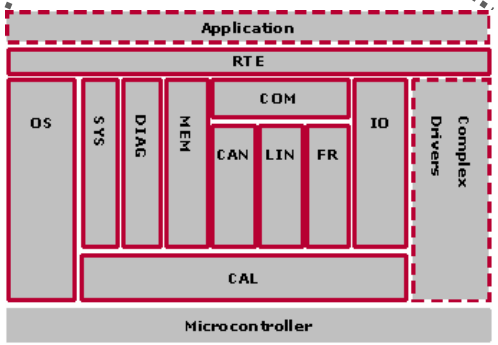
DaVinci Network Designer Network communication design and schedule definition

System Description



Extract of System Description

ECU Software Development



DaVinci Developer

DaVinici Configuration Pro.

Supplier

ECU Configuration Description

Exercise 1:

Match the AUTOSAR key words to their meanings.

1.	SWC		Contents of the ports
2.	SWC mapping		Configuration of the target platform
3.	Data mapping		The smallest logical unit of the application
4.	Target platform		Allocation of data to messages (K-matrix)
5.	Ports		Executable code units of the application
6.	Task mapping		Creation of executable code for ECU
7.	Code generation		Allocation of SWCs to the ECU
8.	Data		Allocation of runnables to tasks
9.	Runnables	5.	Communication interface to other SWCs

大纲

综述和目标

AUTOSAR入门 (introduction)

AUTOSAR方法论 (methodology)

> **AUTOSAR实时环境 (RTE)**

AUTOSAR基础软件 (BSW)

Vector AUTOSAR实现

从CANbedded到AUTOSAR

RTE——餐馆服务员

SW-C 1

SW-C 2

SW-C n

RTE

Input

Output

Communication

Scheduling

BSW

顾客A

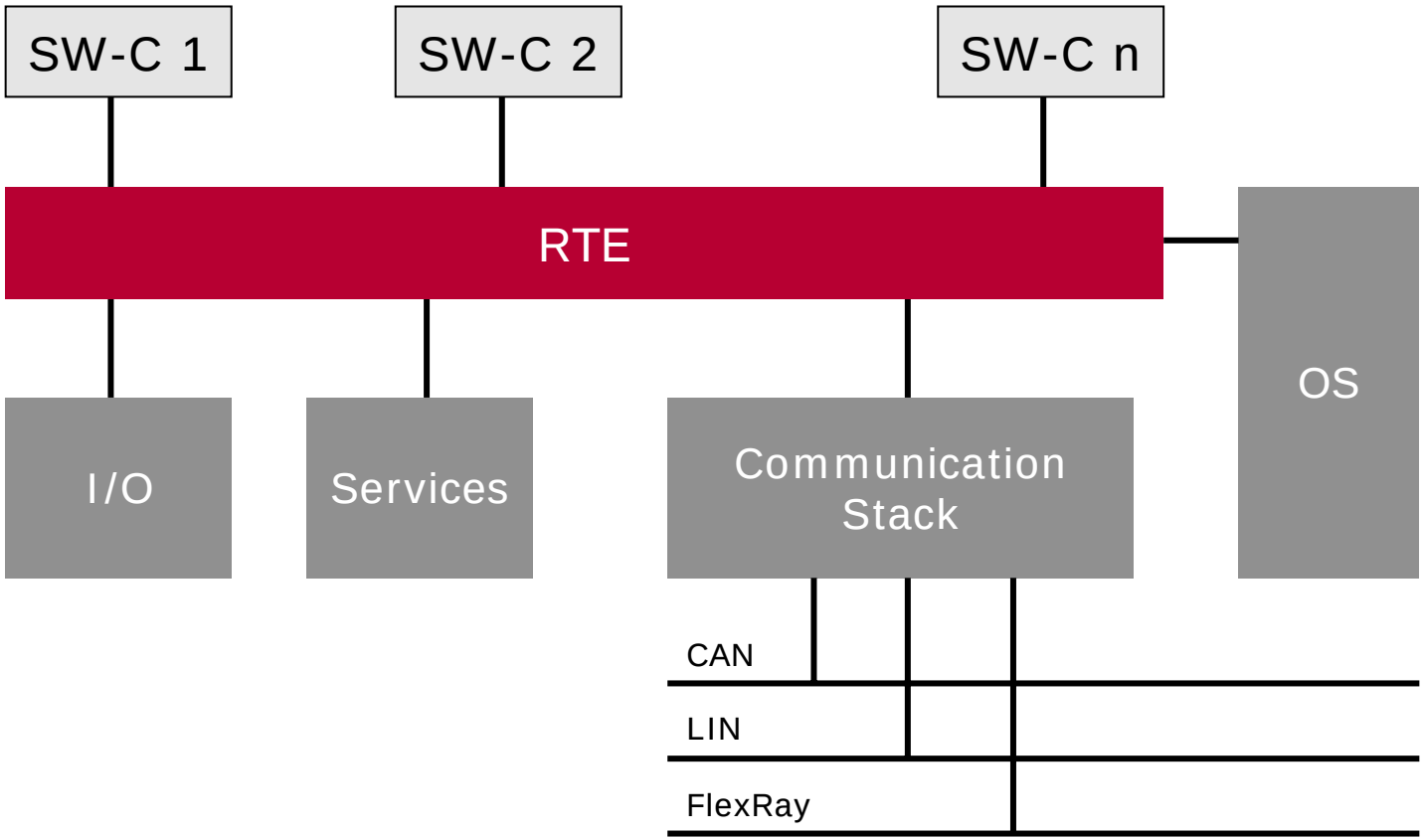
顾客B

帐台

厨师

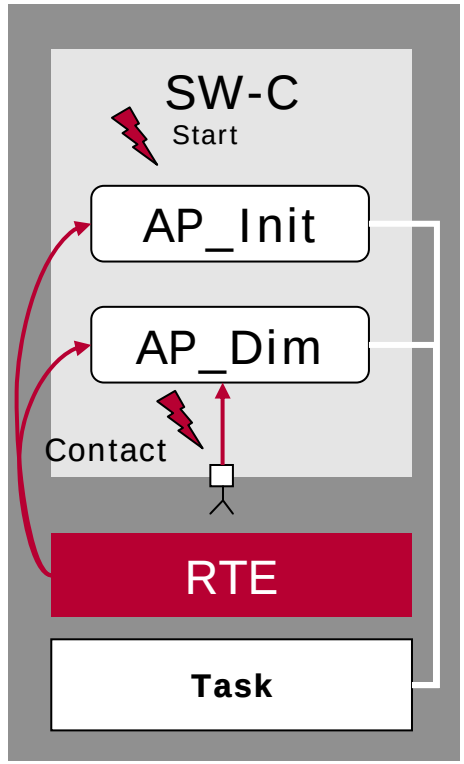
清洁工

RTE——VFB的具体实现



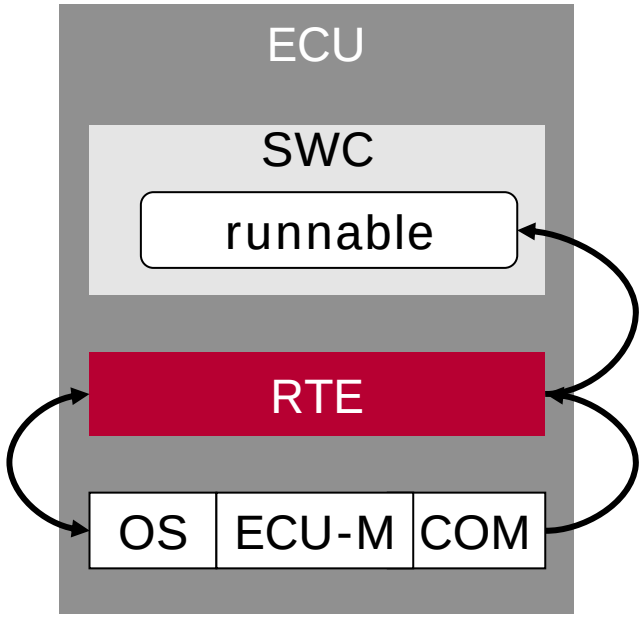
RTE——VFB的具体实现

- RTE需要配置(e.g. 把runnables对应到OS的tasks中去)
- 通过RTE的事件触发runnables的运行
- 生成调用runnables的task代码
- 配置OS的一部分 (tasks, events, alarms)
- 实现SWC之间的通信
- 每个ECU的RTE因SWC的需求而异
- RTE抽象了OS，防止SWC直接访问OS和BSW



RTE——VFB的具体实现

- SWCs with runnables



OS:

```
SetRelAlarm()
ActivateTask()
Schedule()
WaitEvent()
```

SWC:

```
Runnable1(){
  // Code for Runnable
}
```

COM:

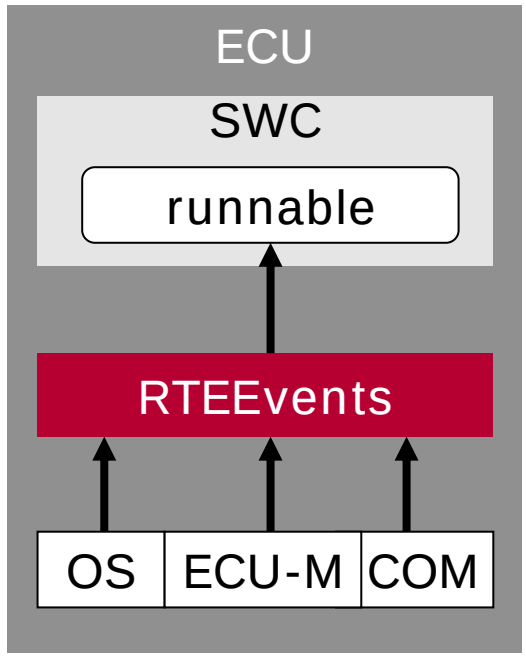
```
Rte_COMCbk_<sigName>
```

ECU-M:

```
Rte_Start(), Rte_Stop()
```

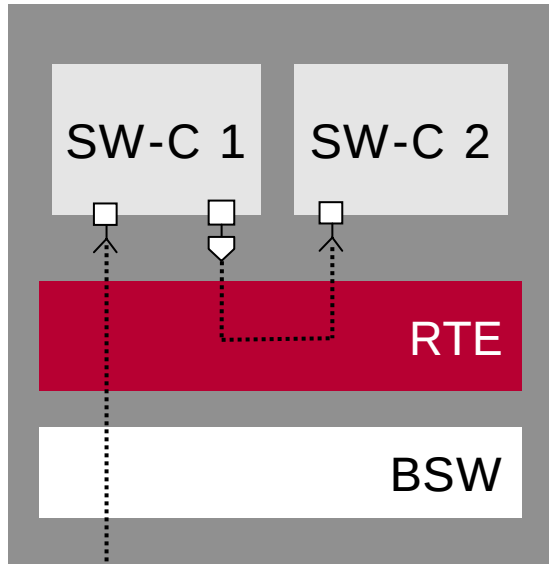
RTE——VFB的具体实现

- Runnables的触发条件
  - 定时时间
    - 周期性触发 (例如使用OS的Alarm)
  - 数据接收事件(S/R)
    - 当收到数据时触发
  - 异步服务调用返回事件(C/S)
  - 操作调用事件(C/S)
  - 数据接收错误事件(S/R)
  - 数据发送完成事件(S/R)
  - 状态切换事件



RTE——VFB的具体实现

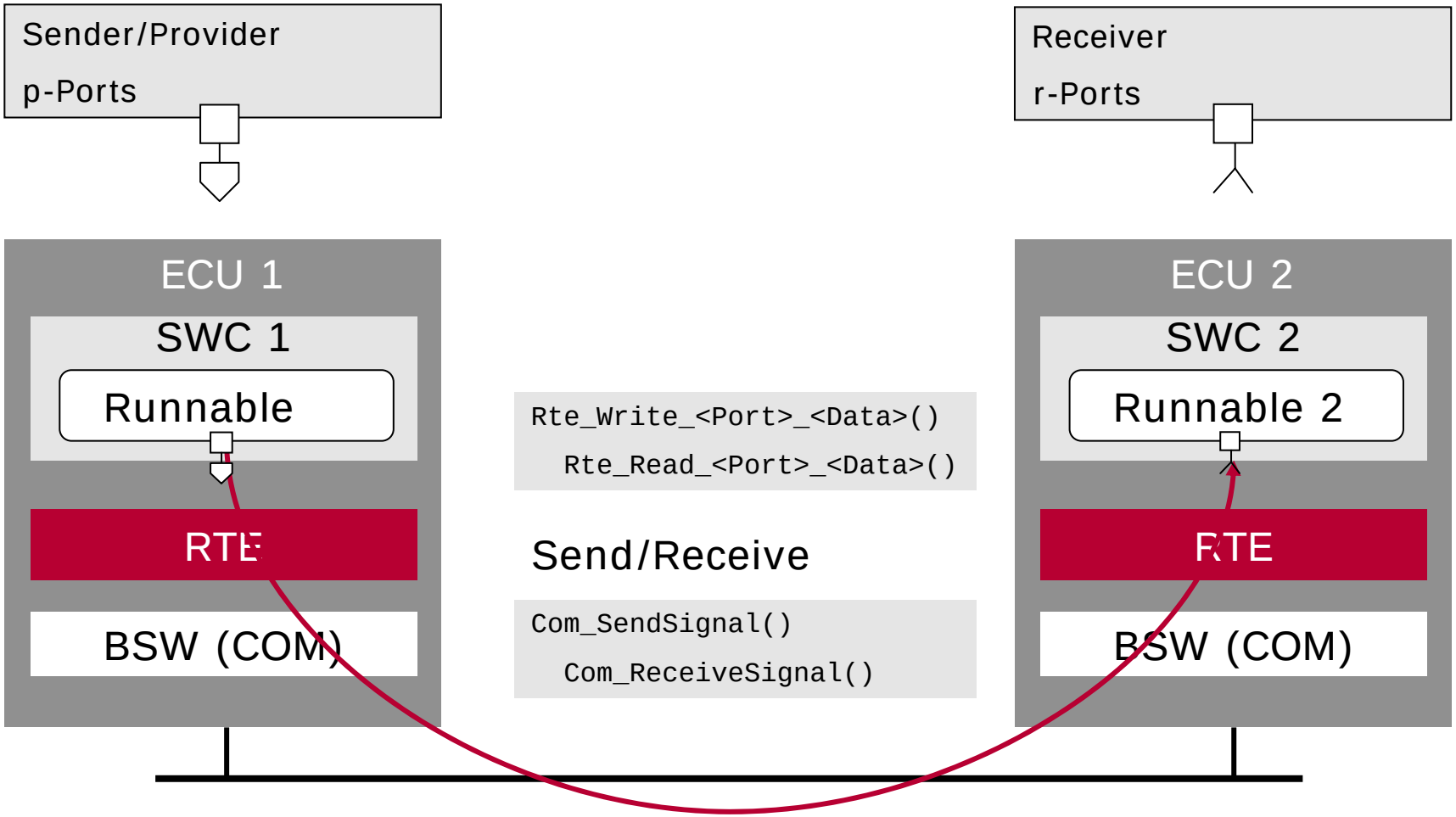
- RTE作为SWC和BSW之间的通信机构
  -> RTE是VFB的实现
  - Sender-Receiver
  - Client-Server
  - Intra-ECU and Inter-ECU (via COM)
  - RTE implements callbacks of AR-COM



- Other features:
  - 保证数据一致性(e.g. exclusive area)
  - 支持简单数据及复杂数据(records)
  - SWC多途径应用

AUTOSAR RTE

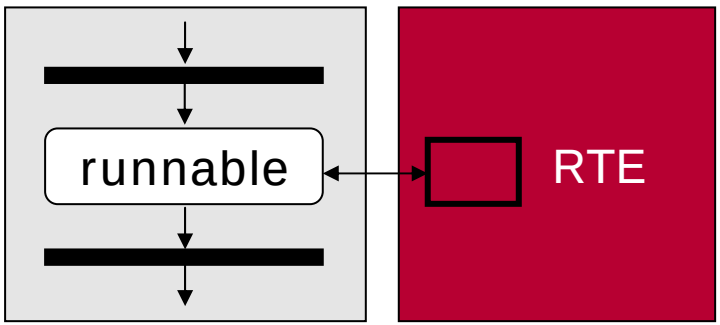
Sender/Receiver 通信-> ECU之间



Sender/Receiver 通信 -> 不使用队列（直接访问）

- **“Last is Best”**
  - RTE直接访问数据地址
  - 1:n通信
  - 初始值即为默认值
  - 适用于实时性要求高的数据

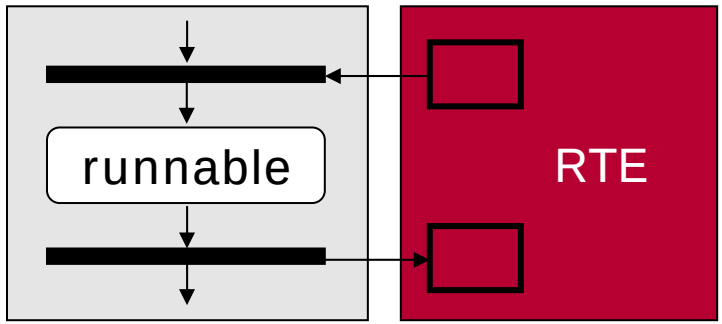
```
Std_ReturnType Rte_Read_<p>_<d> (OUT <DataType> *data)
Std_ReturnType Rte_Write_<p>_<d> (IN <DataType> data)
```



Sender/Receiver 通信 -> 不使用队列（缓存访问）

- **“Last is Best”**
  - 在进入runnable之前RTE为数据建立副本
  - 在runnable运行结束之后RTE把副本数据拷贝到实际数据地址
  - 在runnable运行过程中只操作副本，实际数据不会改变
  - 适用于有一致性要求的数据组

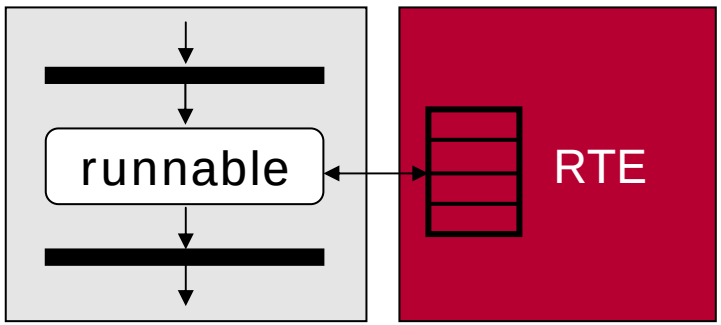
```
<DataType> Rte_IRead_<r>_<p>_<d> (void)
void Rte_IWrite_<r>_<p>_<d> (IN <DataType> data)
```



Sender/Receiver 通信 -> 使用队列

- **“event” (isQueued=True)**
  - “查询接收”或“等待接收”
  - RTE从队列中读取数据
  - “等待接收”有超时处理

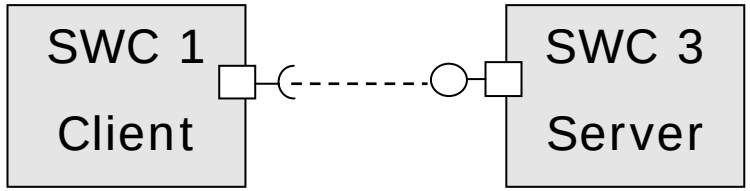
```
Std_ReturnType Rte_Receive_<p>_<d> (OUT <DataType> *data)
Std_ReturnType Rte_Send_<p>_<d> (IN <DataType> data)
```



- **“无效数据元素”**
  - 应用：表示sensor发送的数据无效
  - 不能使用队列
  - 属性“Invalid Value”
  - 设置无效值
    - `Rte_Invalidate_<p>_<d>()`
  - 接收方对无效值的处理
    - 回调函数`Rte_Feedback_<p>_<d>()`
    - `Rte_Read_<p>_<d>()`的返回值为`RTE_E_INVALID`
    - 也可以通过收到无效值来激活runnable的运行

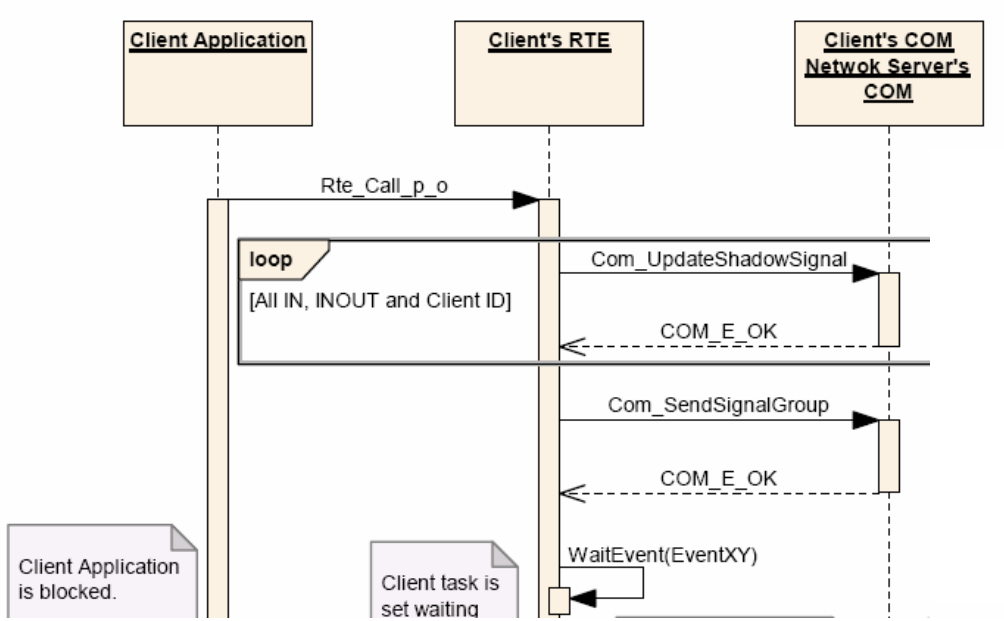
Client/Server 通信

- 通信方式
  - n:1
- 服务调用
  - Client调用Server端操作
  - Server端SWC中的操作一般是runnables
  - 同步/异步调用
  - Server端runnables:
    - 没有分配到Task中（直接被调用）
    - 分配到Task




```
Std_ReturnType  Rte_Call_<p>_<o>([IN | IN/OUT | OUT <param_1>],...
                                 [IN | IN/OUT | OUT <param_n>])
```

- 同步通信
  - 等待Server 端响应(Client 在等待过程中停止)



  - Server runnable
    - `void **GetTime**(uint32 *hour, uint *minute, *uint second);`
  - RTE Client API

```
Std_ReturnType Rte_Call_<Port>_GetTime(uint32 *hour,
                                       uint *minute, uint *second)
```

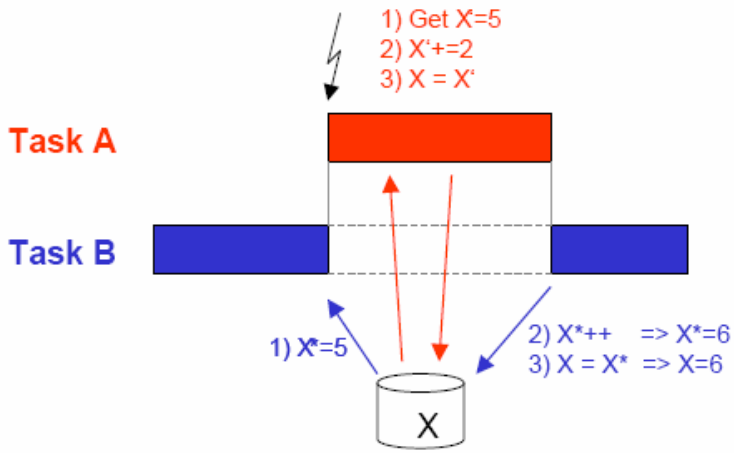
- 异步通信
  - Client不会停止运行 (不等待结果)
  - Client 通过Rte_Result... 获得Server端响应
    - Polling或waiting
    - 超时处理
  - RTE Client API

```
Std_ReturnType  Rte_Result_<p>_<o>([IN/OUT | OUT <param_1>],..
                                   [IN/OUT | OUT <param_n>])
```

  - RTE可以通过接收到响应来激活Client端的runnable

SWC内部通信

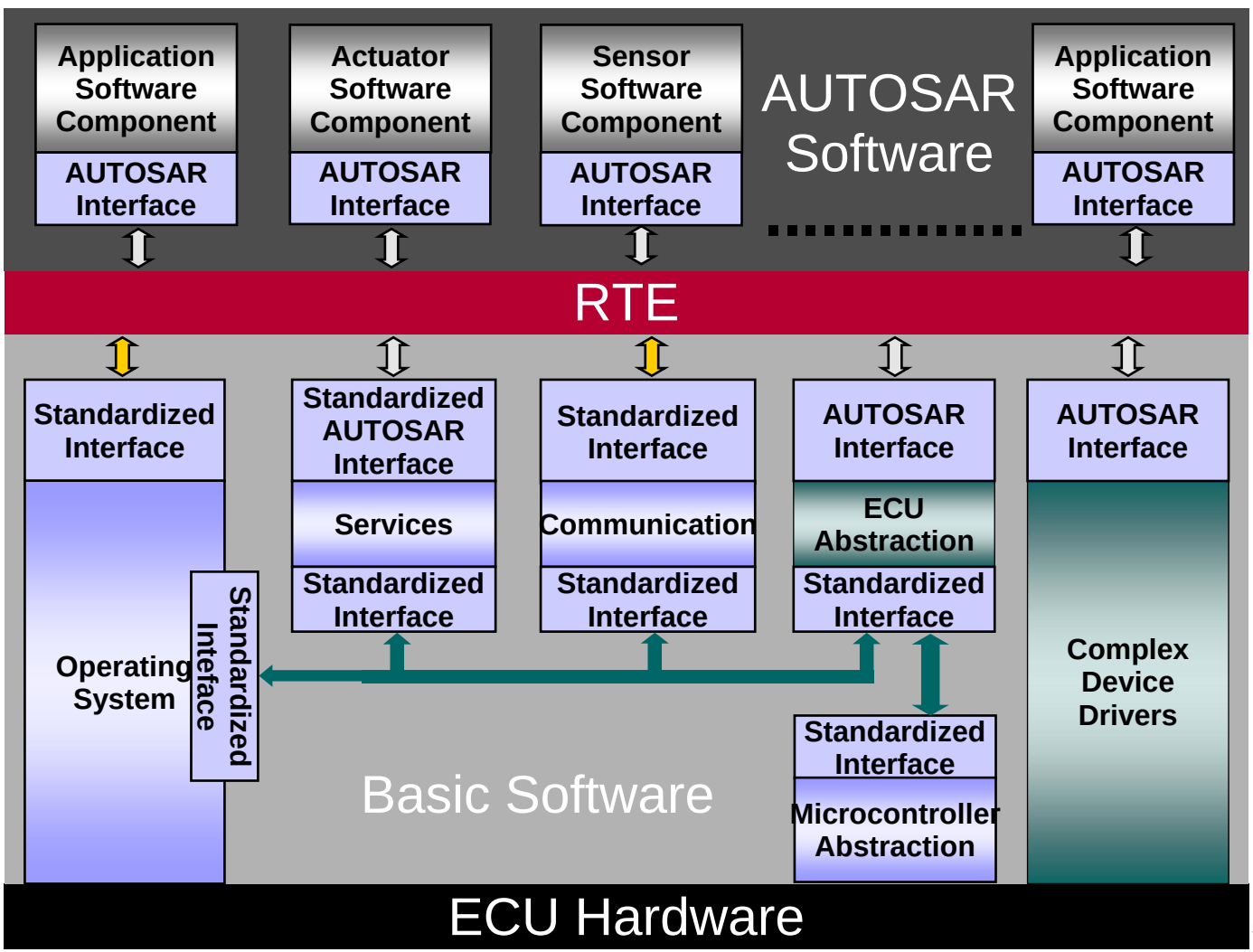
- 问题: 同一个SWC内的、运行在不同Task上的runnable之间的通信.
- 目标: 保证数据一致性
- 解决办法:
  - 专用区域(Exclusive Areas)
    - Entire block or RTE protected
    - `Rte_Enter_<name>()`
  - 内部变量(Inter-runnable variables)
    - Only variable protected
    - `Rte_IrvWrite_<re>_<name>`

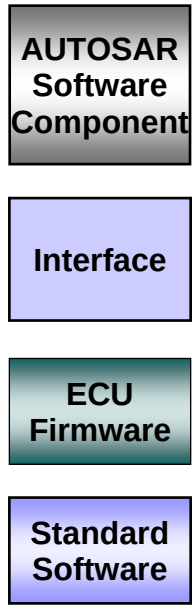


- 注意: 不同SWC之间的通信，无论是ECU内部还是ECU之间，都不会遇到这个问题，因为RTE会负责保证数据一致性.

AUTOSAR RTE

Interfaces



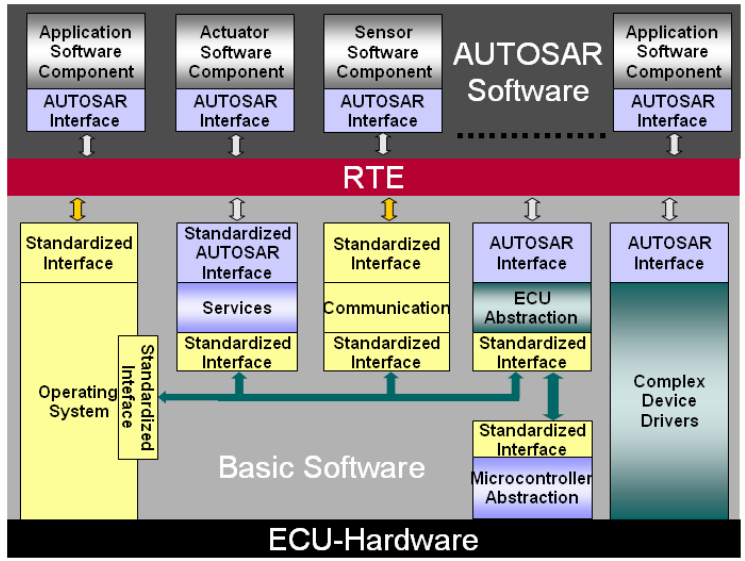


Source: http://www.autosar.org

AUTOSAR RTE

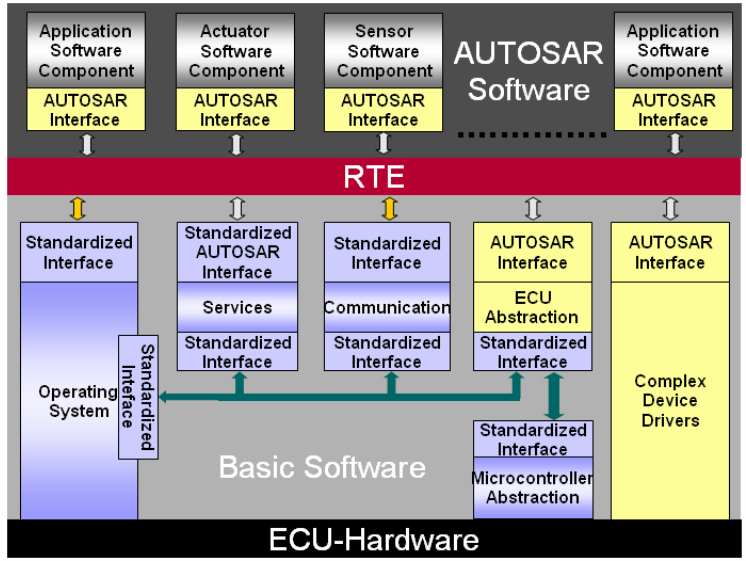
Standardized Interface

- C-API
- OS和COM只能通过RTE间接访问
- 没有Ports
- OS: RTE调度tasks和使用OS API
  - `Schedule(), WaitEvent()`
- COM: RTE统筹COM的callbacks和API
  - `Com_SendSignal()`
- EcuM: `Rte_Start(), Rte_Stop()`



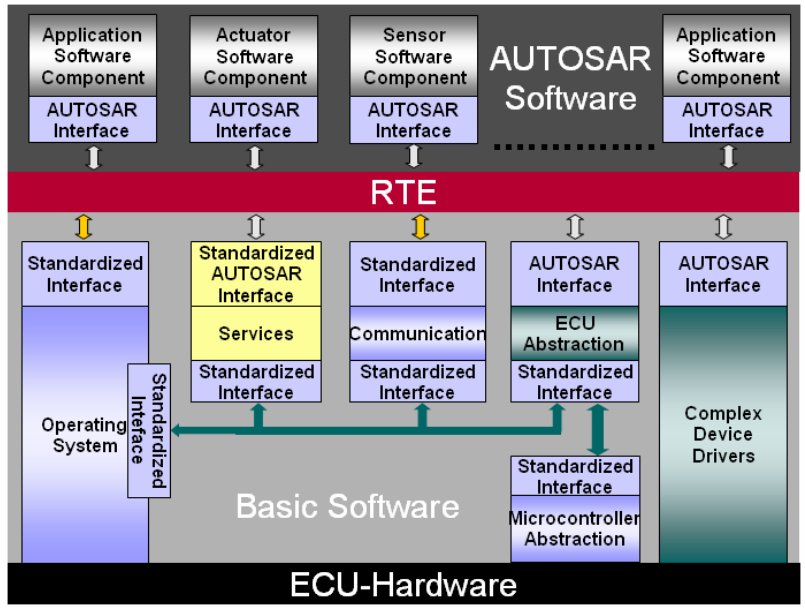
AUTOSAR Interface

- I/O硬件接口和SWC一致:
  - Ports
  - Runnables
- ECU Abstraction = “Firmware”, 只能被sensor/actuator SWC访问
- API:
  - S/R -> Rte_Write_<p>_<d>
    - **Rte_Write_DimmLight_DimmValue(value)**
  - C/S -> Rte_Call_<p>_<o>



Standardized AUTOSAR Interface

- 标准的与ECU无关(e.g. ECU Abstraction)
- RTE -> BSW特定“**Service Ports**”
- API:
  - 某些特殊SWC的通信模式



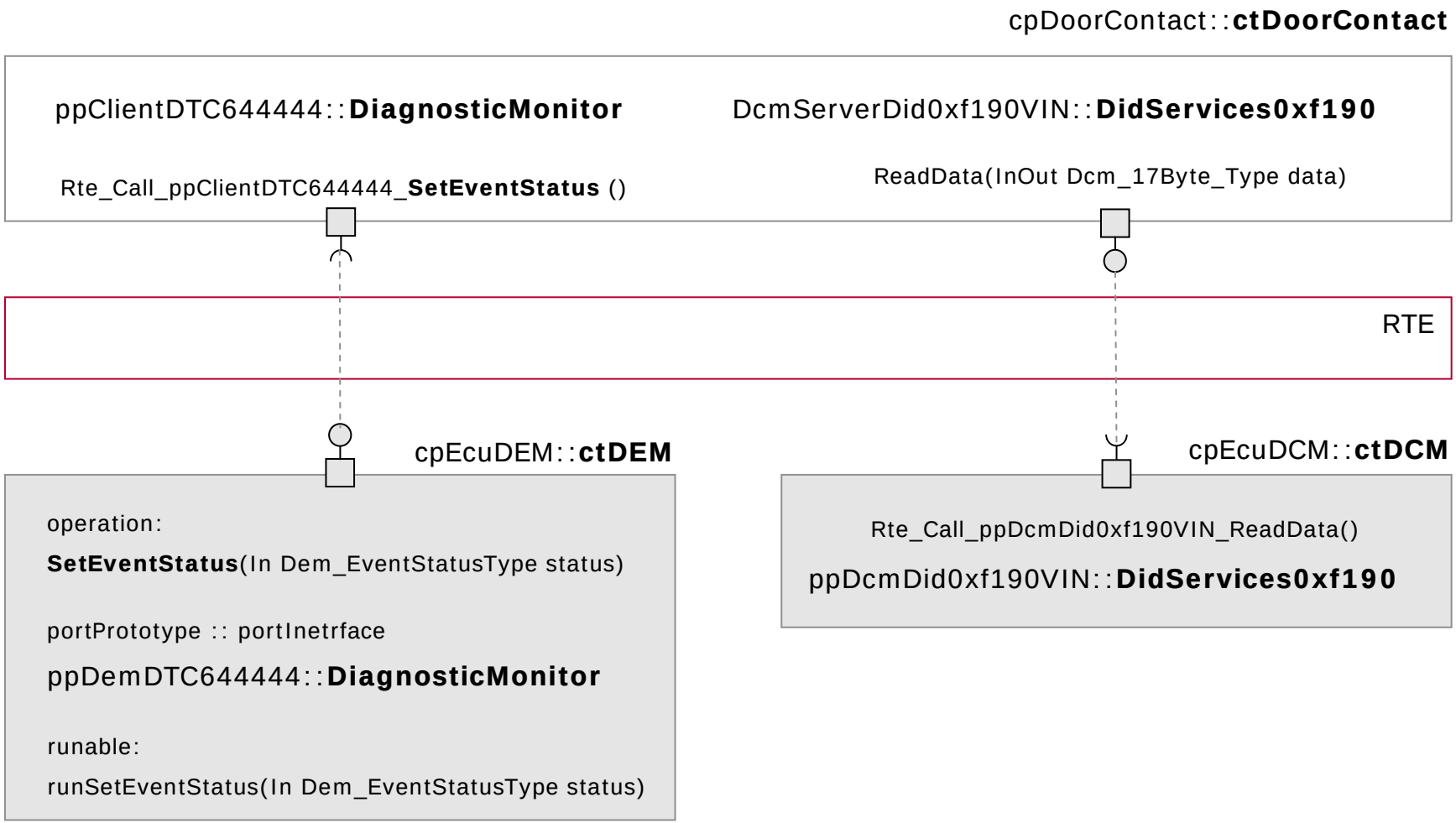
  - SWC -> RTE

    `Rte_Call_<portPrototype>_<operation>()`

  - <Operation> DEM:

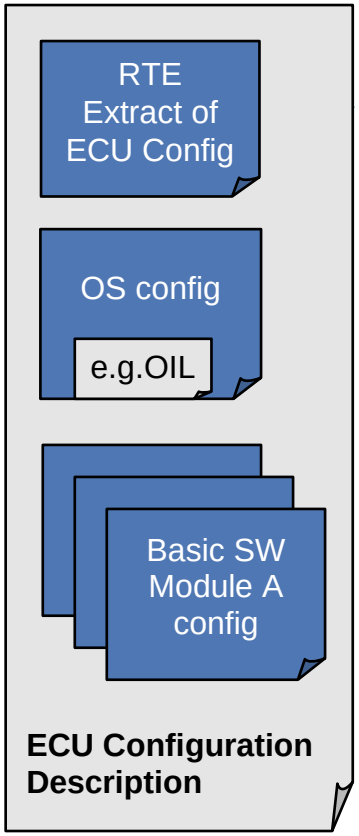
    `SetEventStatus (In Dem_EventStatusType status)`

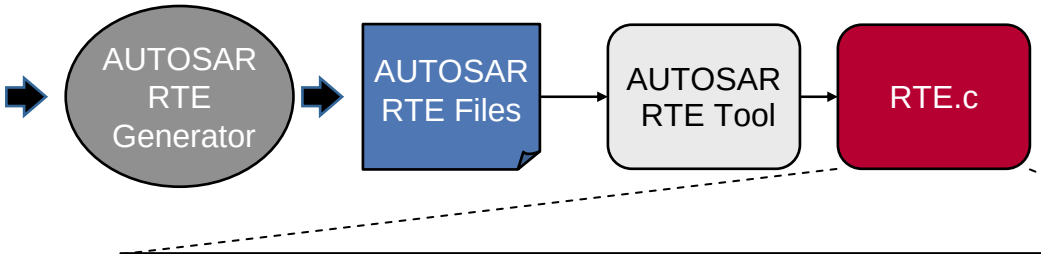
Standardized AUTOSAR Interface



AUTOSAR RTE

生成阶段






```
TASK(ControlTask)
{
    EventMaskType ev;

    for(;;)
    {
        (void) WaitEvent(Rte_TE_AP_Dimmer | Rte_TE_AP_DoorContacts);

        (void) GetEvent(ControlTask,&ev);
        (void) ClearEvent(ev);
        if(ev & Rte_TE_AP_Dimmer)
        {
            AP_Dimmer(NULL_PTR);
            (void) Schedule();
        }
        if(ev & Rte_TE_AP_DoorContacts)
        {
            AP_DoorContacts(NULL_PTR);
            (void) Schedule();
        }
    }
}
```

Exercise 2

判断下列陈述正确与否:

1. RTE替代了从前的基础软件。

   答:

2. 如果调用了函数Rte_Write_<p>_<d>()，则数据会等到runnable运行结束之后才被发送。

   答:

3. 每个ECU都需要生成不同的RTE。

   答:

4. RTE仅仅处理了SWC之间的通信，包括ECU内部和ECU之间。

   答:

5. 如果SWC从一个ECU移动到另外一个ECU，则调用的通信函数（如RTE_Write）也会发生变化。

   答:

6. 服务器的一个操作可以被多个客户端调用。

   答:

大纲

综述和目标

AUTOSAR入门 (introduction)

AUTOSAR方法论 (methodology)

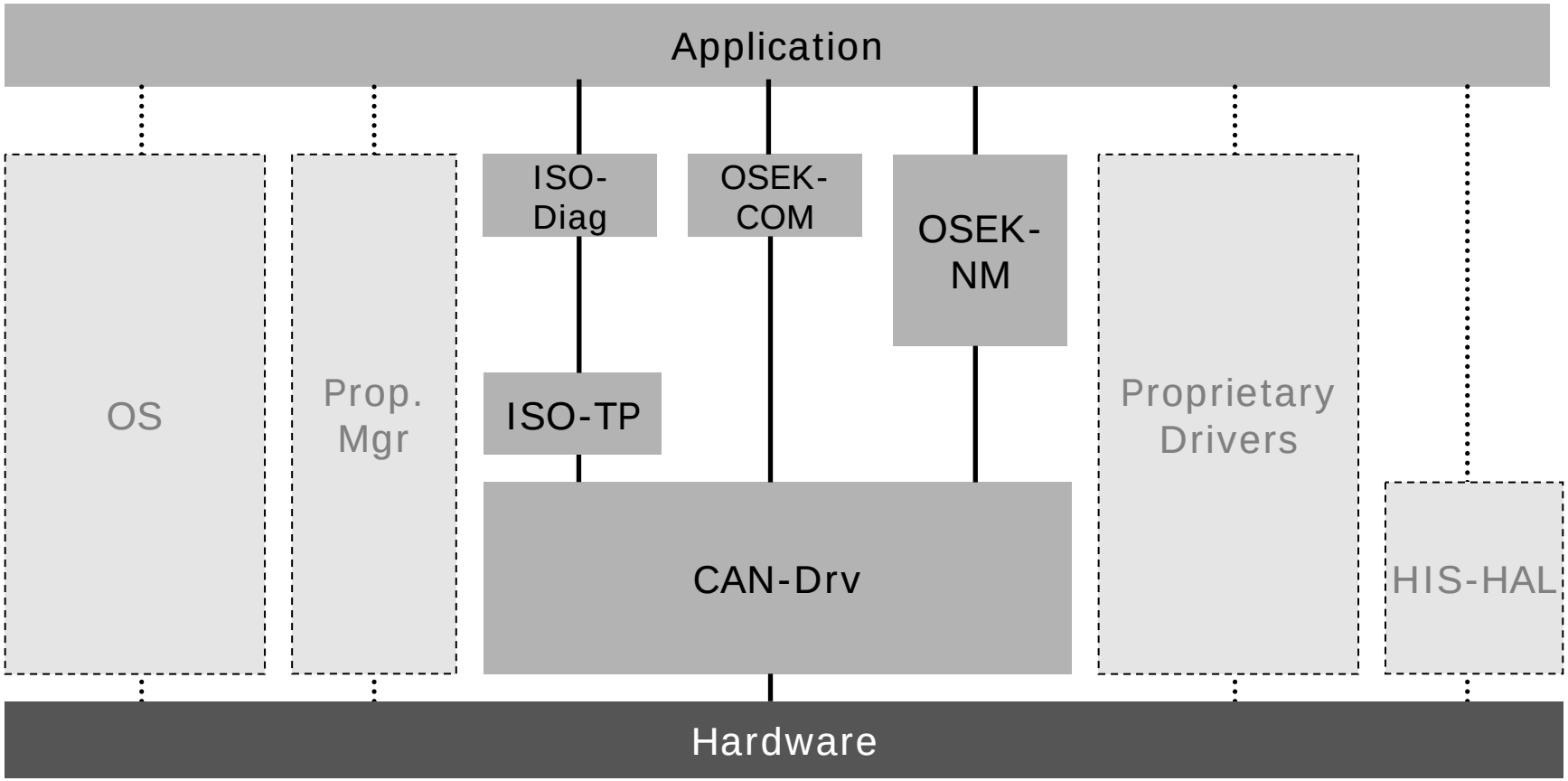
AUTOSAR实时环境 (RTE)

> **AUTOSAR基础软件 (BSW)**

Vector AUTOSAR实现

从CANbedded到AUTOSAR

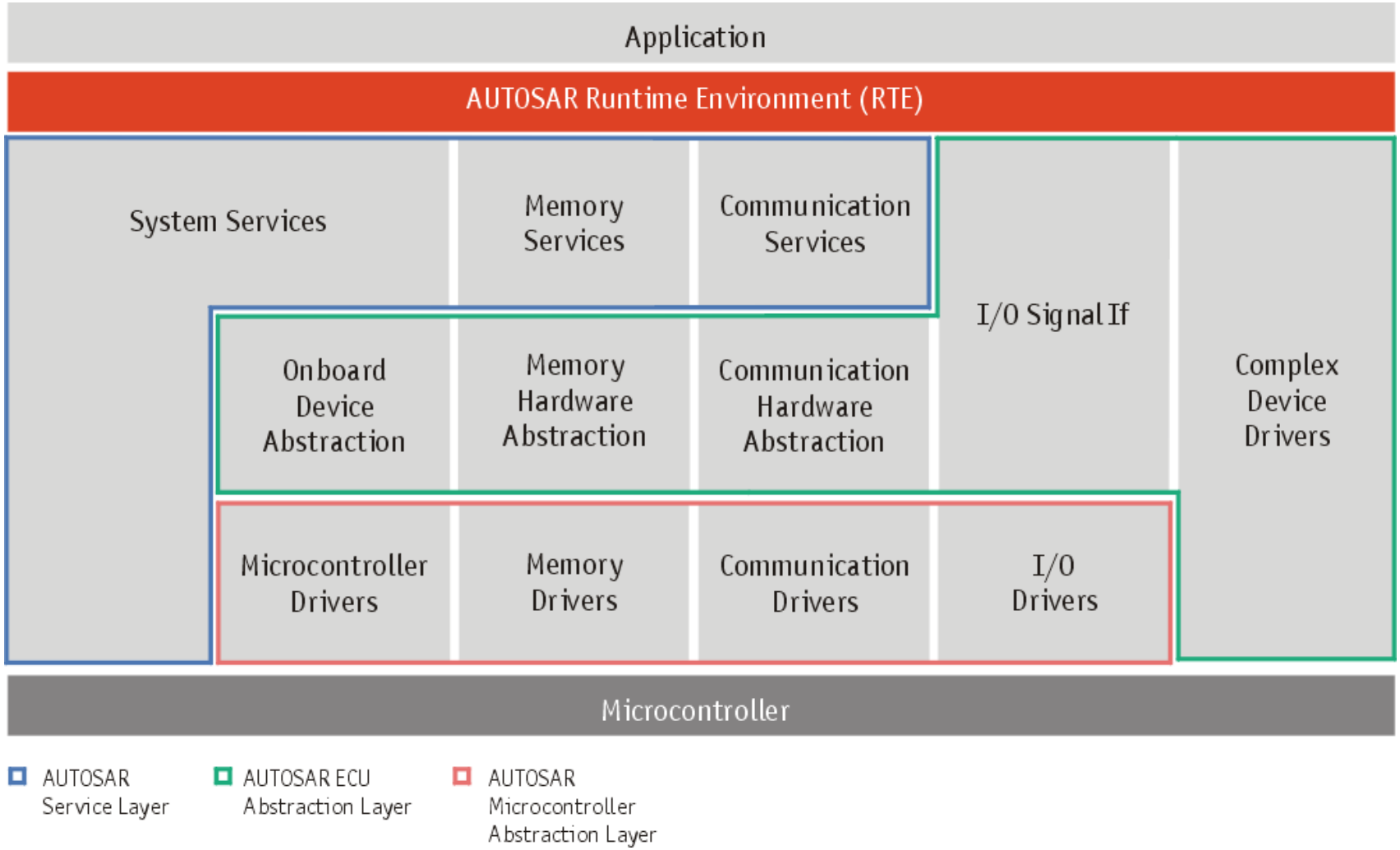
当前使用CAN的ECU软件架构



- 抽象程度低
- 基础软件模块数量少

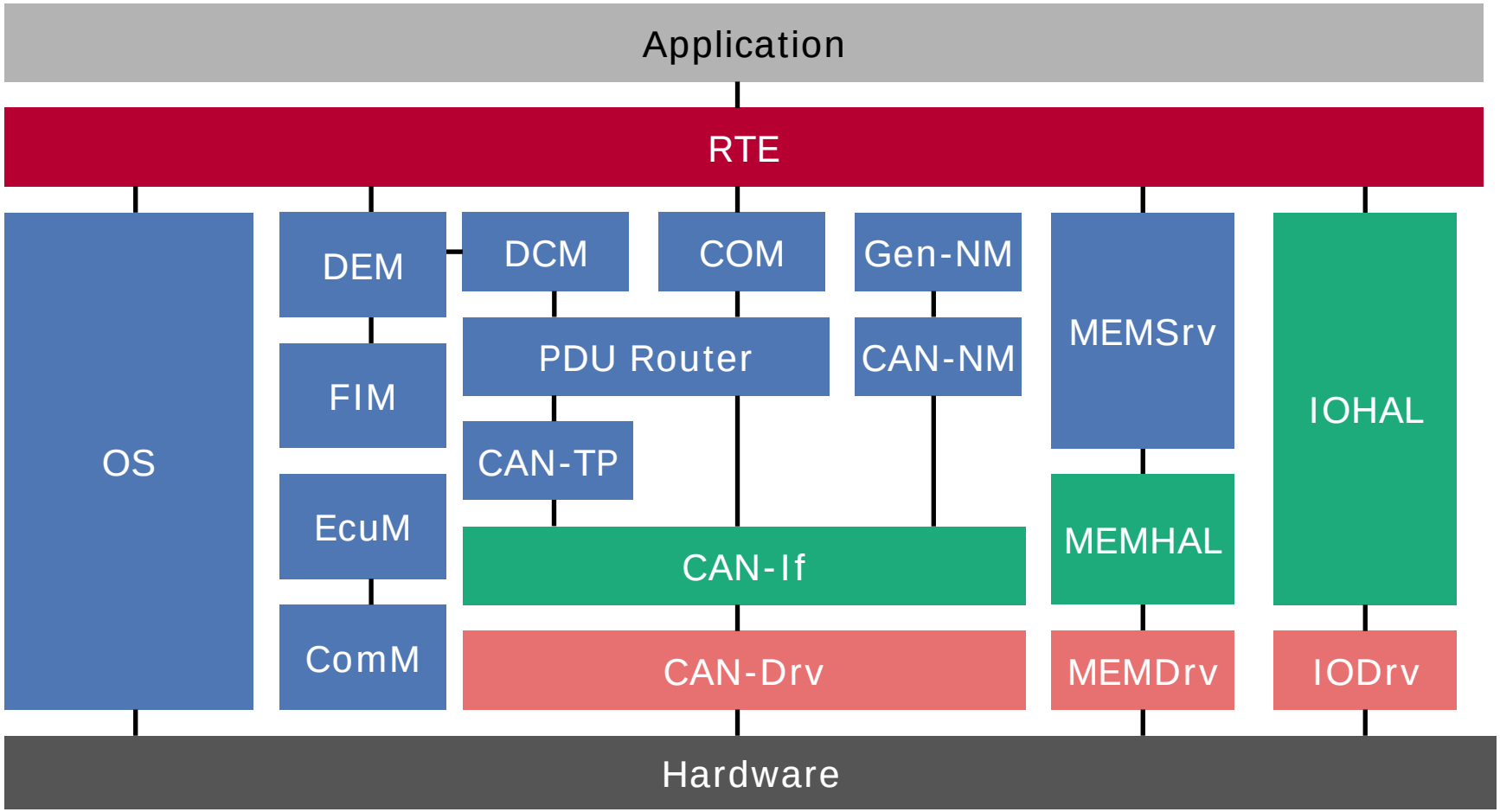
AUTOSAR BSW

AUTOSAR总体架构



AUTOSAR BSW

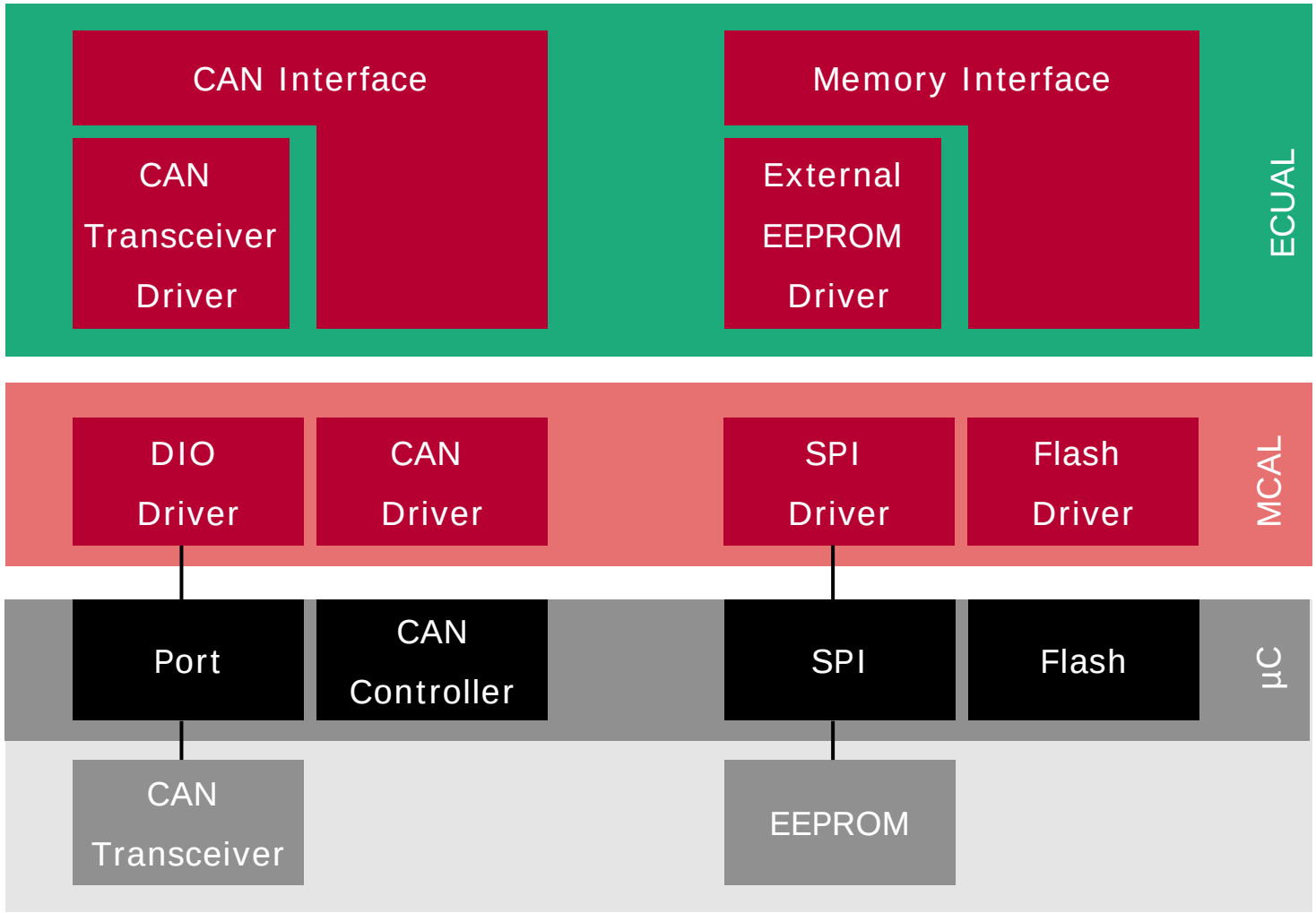
AUTOSAR架构中CAN ECU的软件结构



- 抽象程度高
- 有许多基础软件模块可以使用

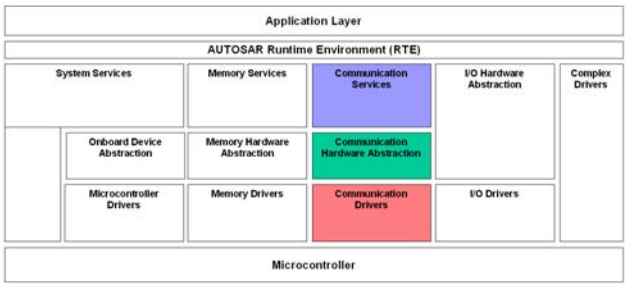
AUTOSAR BSW

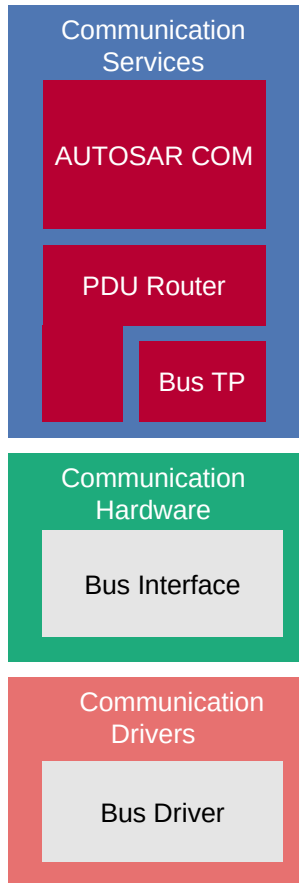
实例



通信

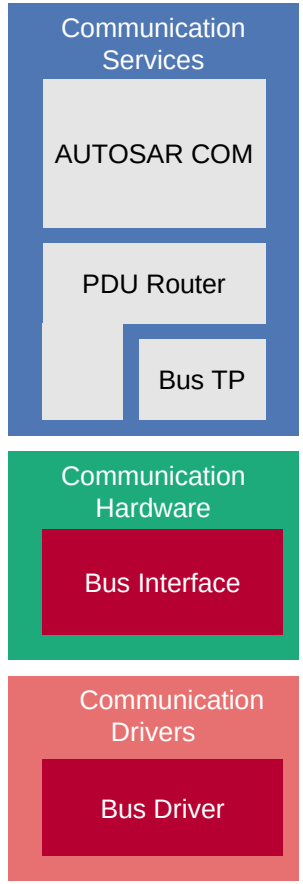
- COM
  - 信号接口
  - 设置/分析PDU
  - 信号网关
- PDU Router
  - 抽象了总线系统
  - PDU网关
- Transport Protocol
  - 数据分包传输



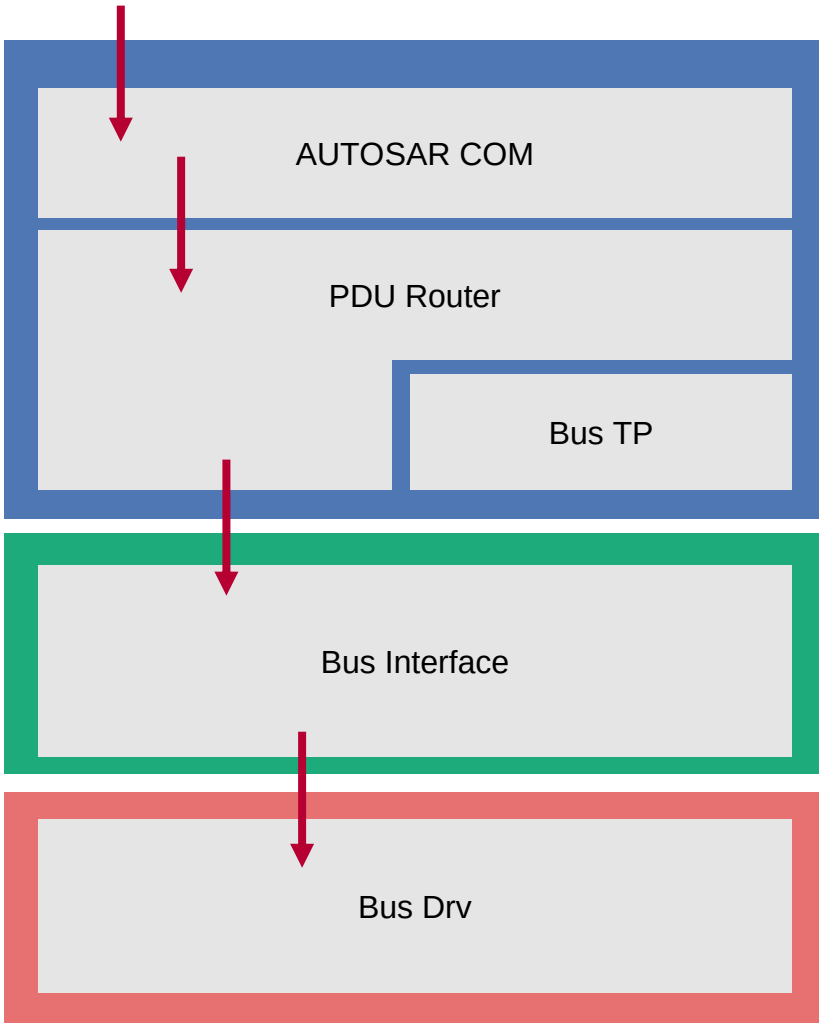


通信

- Bus(CAN/LIN/FlexRay) Interface
  - 与硬件无关，定义每种总线特定的功能
    - 收发队列
    - 组帧 (FlexRay)
    - 管理时间触发总线的调度表 (LIN, FlexRay)
- Driver
  - 硬件相关的操作
    - 初始化，填充buffer，实现ISR
- Transceiver
  - 收发器的操作



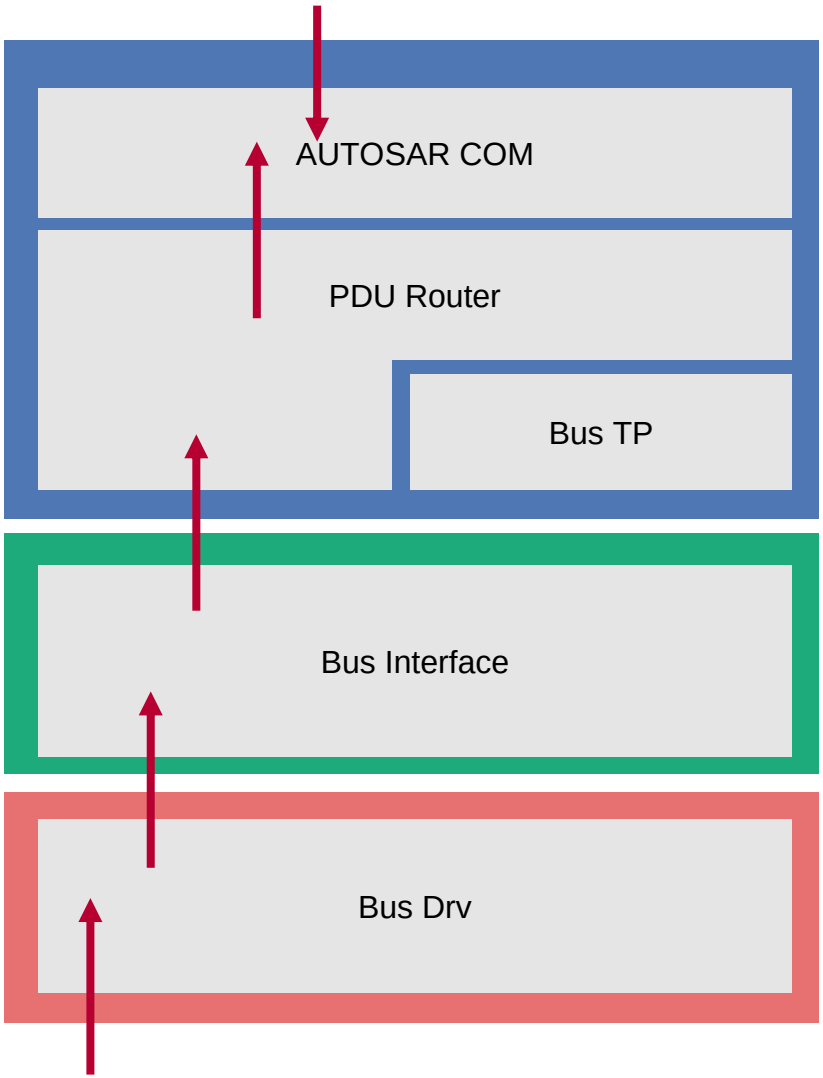
Communication: Sending controlled by COM



- e.g. CAN (spontaneous + cyclic), FlexRay (dynamic segment)
- 写signal
  - 写入PDU buffer
- PDU被PDU Router立刻发送或按周期发送
  - 每个PDU都有一个独立的ID
- PDU Router辨认总线种类，并把PDU发向不同的下级模块
- Interface根据不同的通道，把报文写入不同的队列
- Driver根据报文的优先级立刻发送报文

AUTOSAR BSW

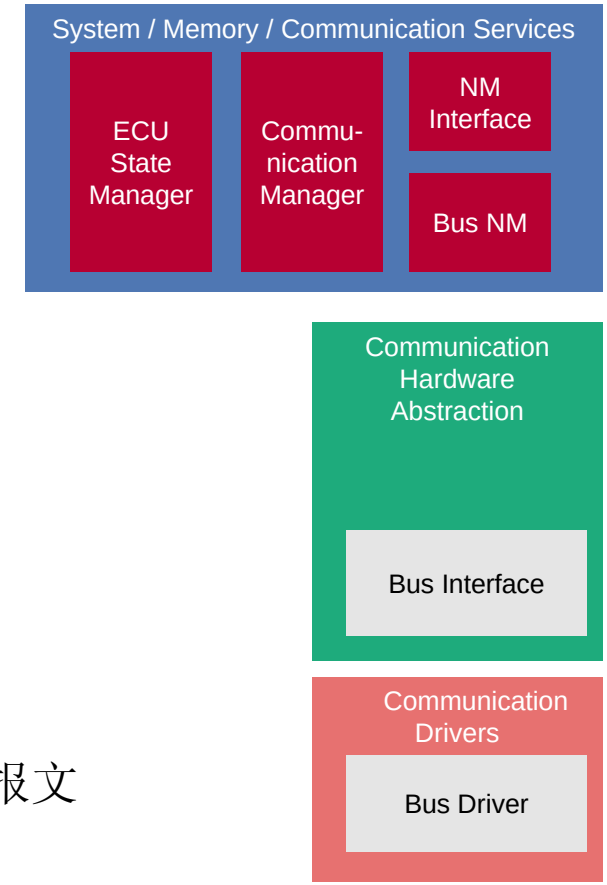
Communication: Receiving



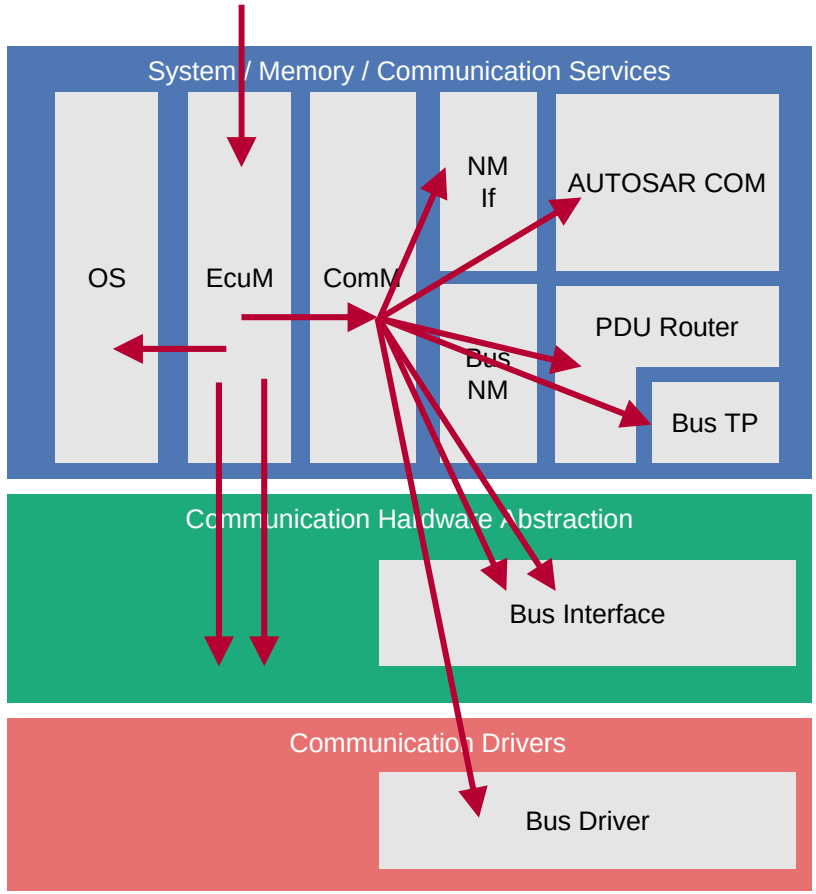
- Interrupt 或 polling, 这里是interrupt
- 接收报文
- 由driver发出Rx中断（函数）
- 通过RxIndication，数据被传递到Interface
- 传递到PDU router
- 传递到COM
  - 如果SWCs使用Data Reception Trigger，就通知RTE
  - 否则存储在buffer中
- 信号被RTE读取

Mode Management

- ECU State Manager
  - ECU 状态机
  - 收集和验证唤醒事件
  - 协调startup/shutdown
- Communication Manager
  - 抽象的总线状态机
  - 通信的Startup/shutdown
- Network Management
  - 具体的总线状态机
  - 如果ECU需要保持工作状态，则周期性发送NM报文

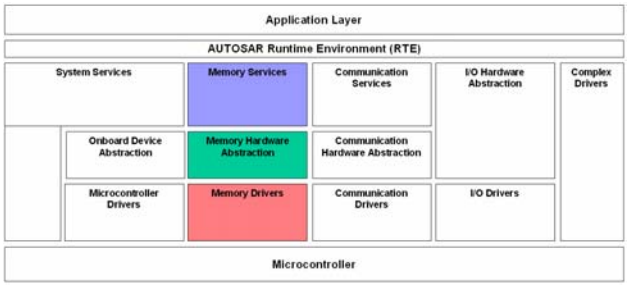


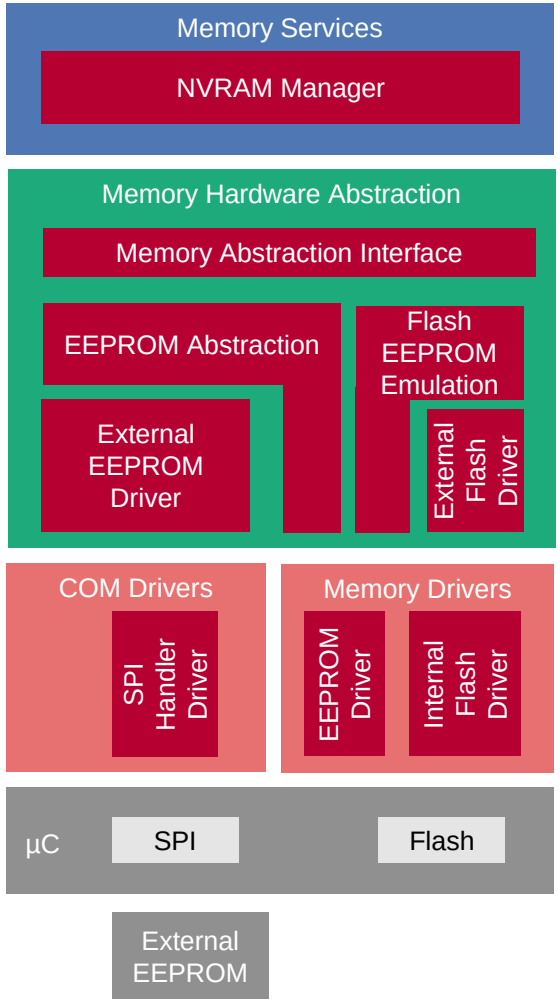
- ECU startup
  - 在ECU启动代码中调用ECUM模块
  - 在OS启动前首先进行硬件的初始化
  - 启动OS
  - OS启动后进行另一部分硬件的初始化
  - 初始化ComM
    - 初始化通信模块
    - 初始化网络管理



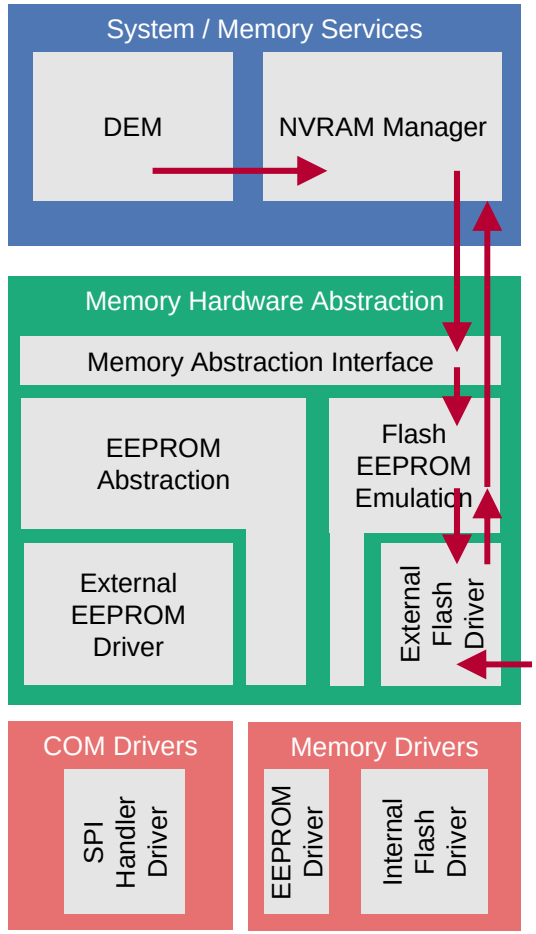
Memory Services

- 目的:
  - 抽像内部（片内）或者外部（片外）存储设备
  - 对内部或外部存储设备（如EEPROM）采用同样的操作方式
- NVRAM manager
- Memory abstraction interface
- Memory drivers



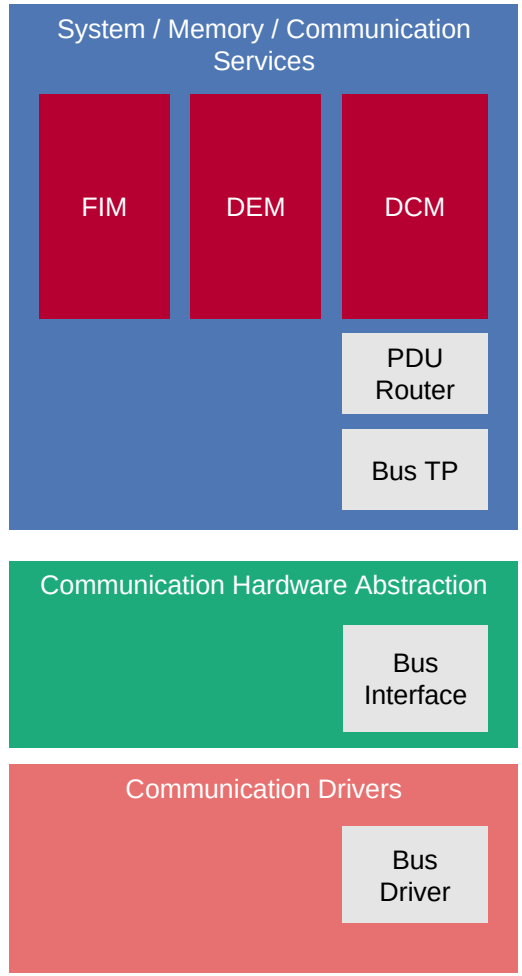


- 将诊断快照写入NVRAM
  - 按照Block-ID写入
  - 写入特定地址
    - 在NVRAM Manager中不care存储器类型
  - 必须通过NVRAM才可以访问Memory Abstraction
  - 既可访问到内部存储器也可以访问到外部存储器
  - 写完后有回调函数通知Memory Abstraction
  - Abstraction层再通知NVRAM Manager



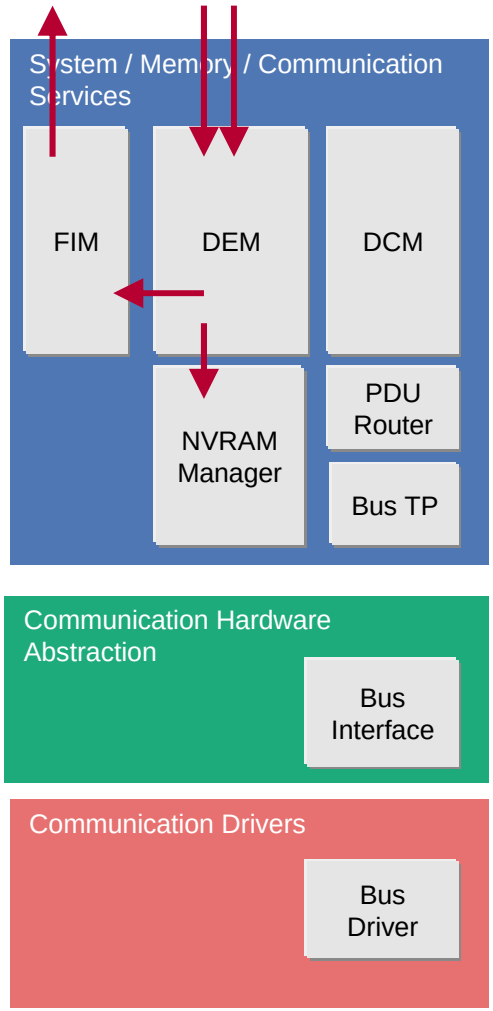
Diagnostics

- Diagnostic Communication Manager (DCM)
  - 实现诊断通信协议
- Diagnostic Event Manager (DEM)
  - 故障存储
    - 记录诊断事件
- Function Inhibition Manager (FIM)
  - 有条件地使能SWC
    - 取决于故障存储内容 (诊断事件)



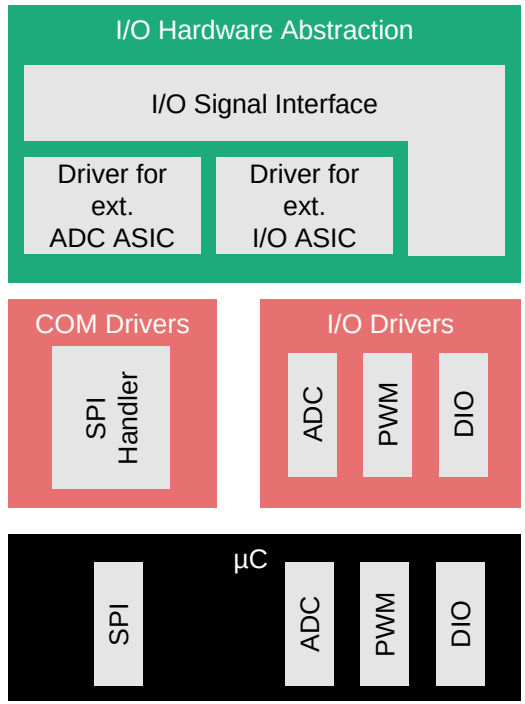
Diagnostics

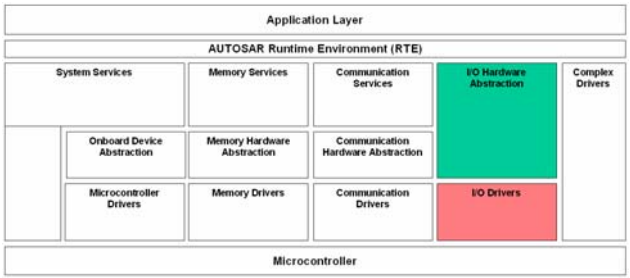
- 检测错误状态并存储故障
  - 保存快照
  - 错误与事件相关联
  - 写相关的非易失性存储器
- 出现错误后关闭部分功能模块
  - 当出现某些错误后DEM会通知FIM
  - 一旦出现错误，SWC可以通过以下方式得知：
    - 回调函数
    - 由SWC查询



Hardware I/O

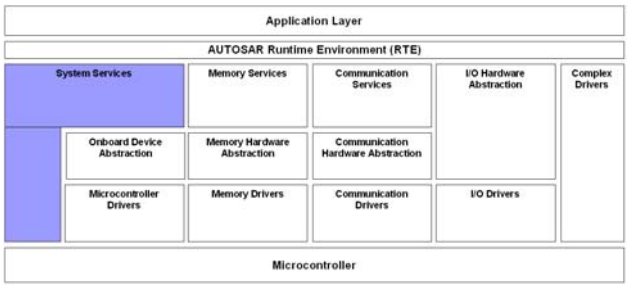
- 抽象（片内/片外）I/O类外部设备
- 无法抽象传感器/执行器设备
  - I/O 信号接口
    - 有条件的读取输入信号
      - 如：信号的消抖
  - 驱动
    - ADC, DIO, PWM





OS

- 基于OSEK/OS规范
- 4个等级分类
  - SC1 (OSEK/OS++)
    - 包含标准OSEK OS标准，除此之外还定义了标准的计数器接口和轮询式的调度表
  - SC2 (OSEKtime)
    - 为SC1提供了时间保护，也就是说，当一个任务执行时间过长，它会被停止；SC2同时还定义了时间监控



  - SC3 (ProtectedOSEK)
    - 包含内存保护
  - SC4 (SC2 + SC3)
    - 同时包含时间保护和内存保护

Exercise 3

Name the layers associated with the following properties:

- ..... offers an API for access to peripherals and devices regardless of their location (µC internal/external) or their connection to the µC (port pins, type of interface)
- ..... contains drivers, which are software modules with direct access to µC-internal peripherals and memory mapped µC-external devices.
- The ..... is a middleware layer providing communication services to the application; it contains AUTOSAR software components and AUTOSAR sensor/actuator components.
- While access to I/O signals is covered by the ....., the ..... offers operating system services, vehicle network communication and management services, memory services, diagnostic services, ECU state management

Source: AUTOSAR_LayeredSoftwareArchitecture.pdf

大纲

综述和目标

AUTOSAR入门 (introduction)

AUTOSAR方法论 (methodology)

AUTOSAR实时环境 (RTE)

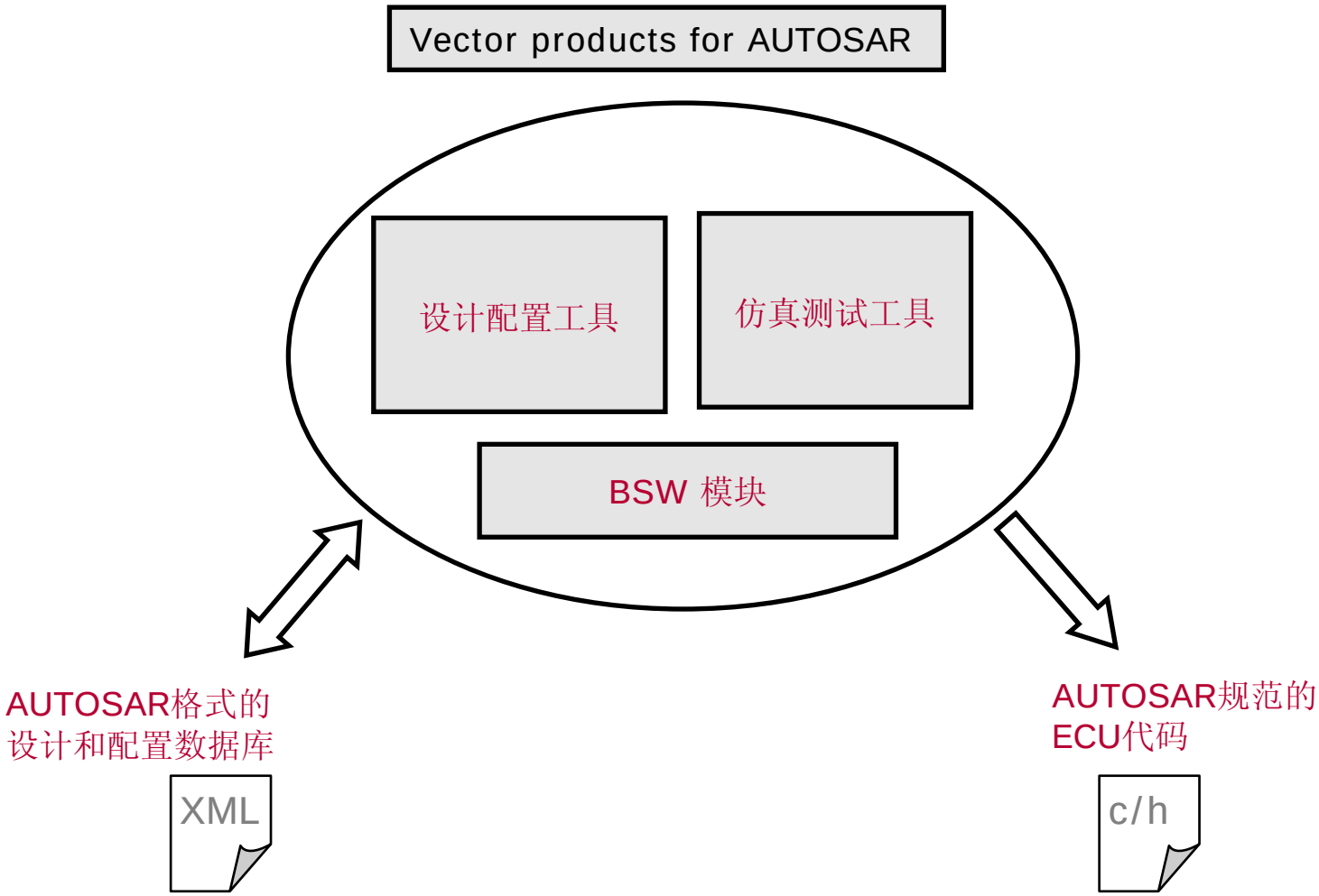
AUTOSAR基础软件 (BSW)

> **Vector AUTOSAR实现**

从CANbedded到AUTOSAR

AUTOSAR 的实际应用

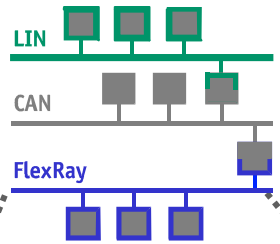
Vector解决方案



Vector 解决方案

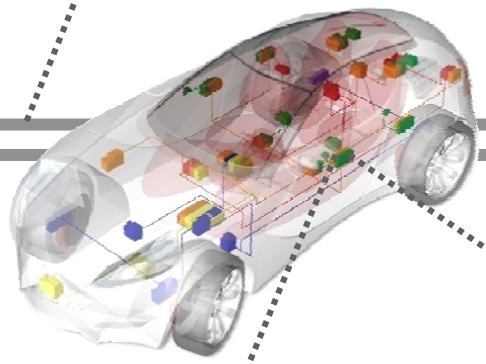
System Design

OEM



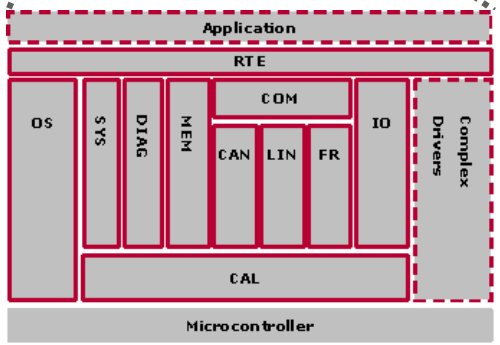
DaVinci Network Designer 网络通信设计和调度表定义

System Description



Extract of System Description

ECU Software Development



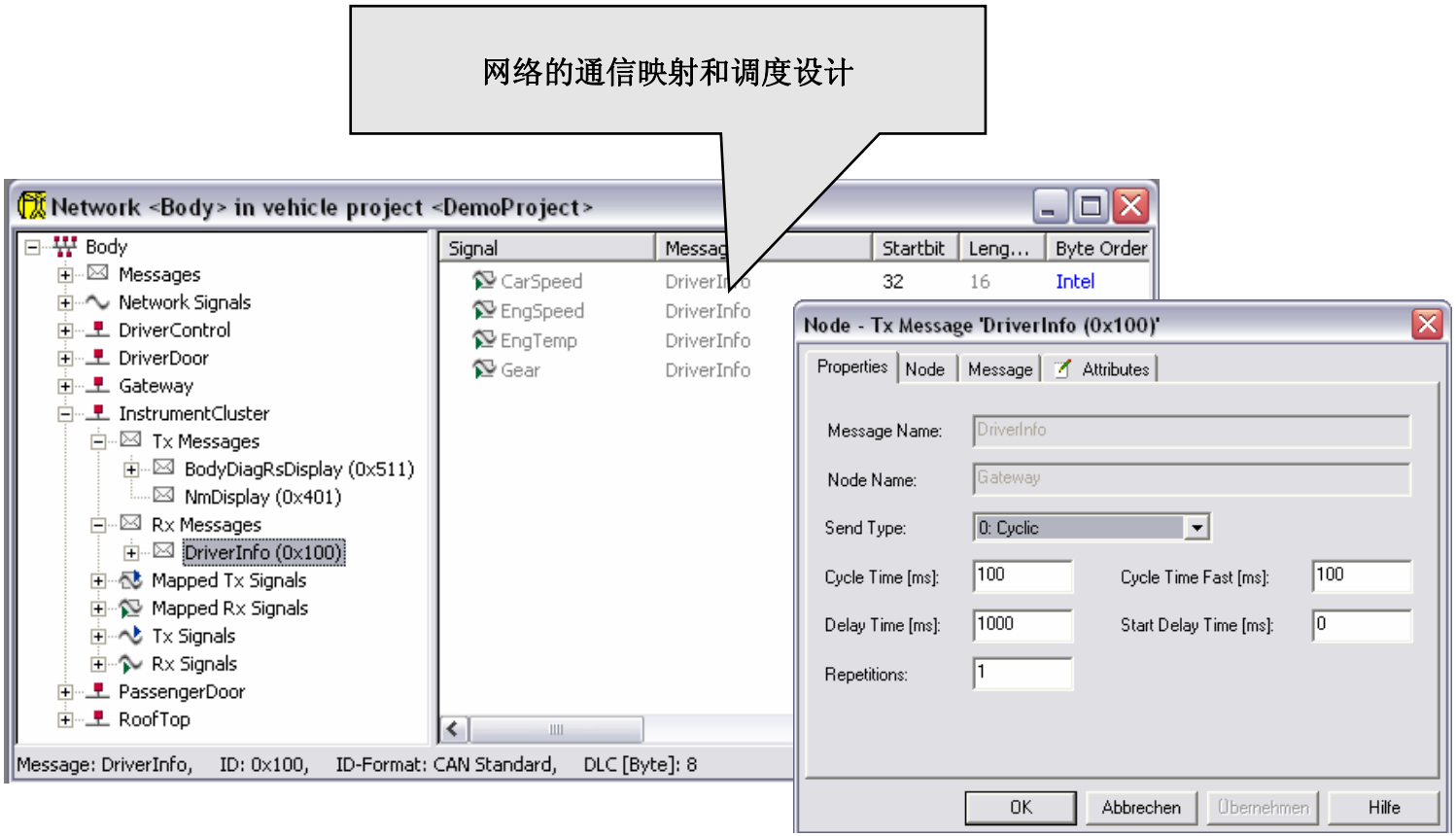
DaVinci Developer ECU配置和RTE生成

DaVinci Configurator Pro BSW配置和生成工具

Supplier

ECU Configuration Description

Tool: DaVinci Network Designer

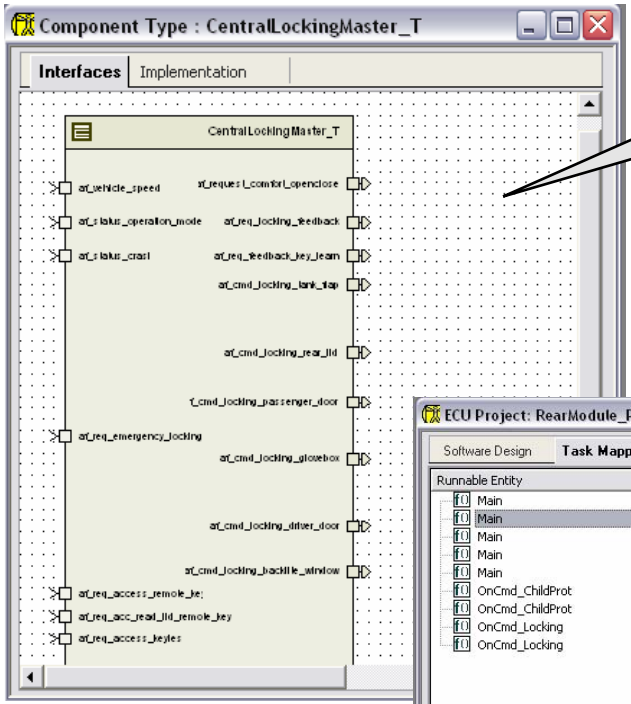


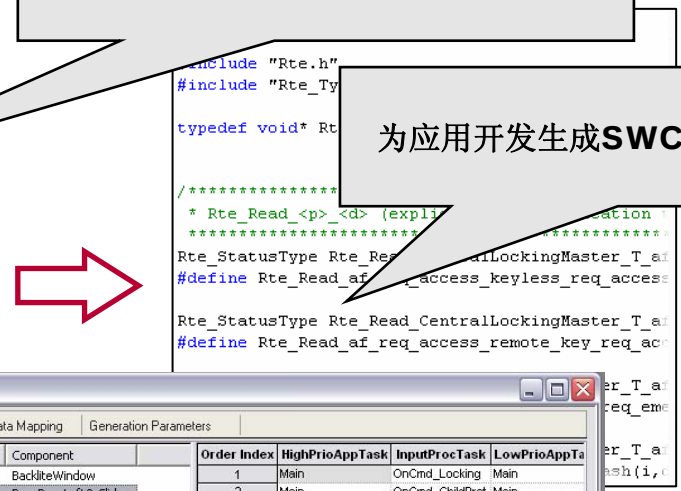
ECU 配置

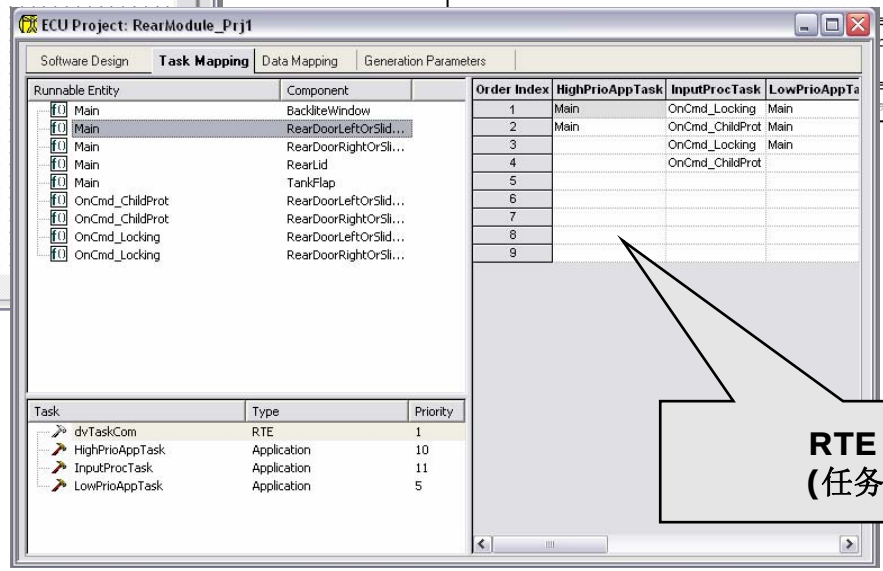
Tool: DaVinci Developer

图形化编辑SWC的接口和内容

为应用开发生成SWC的头文件和内容模板

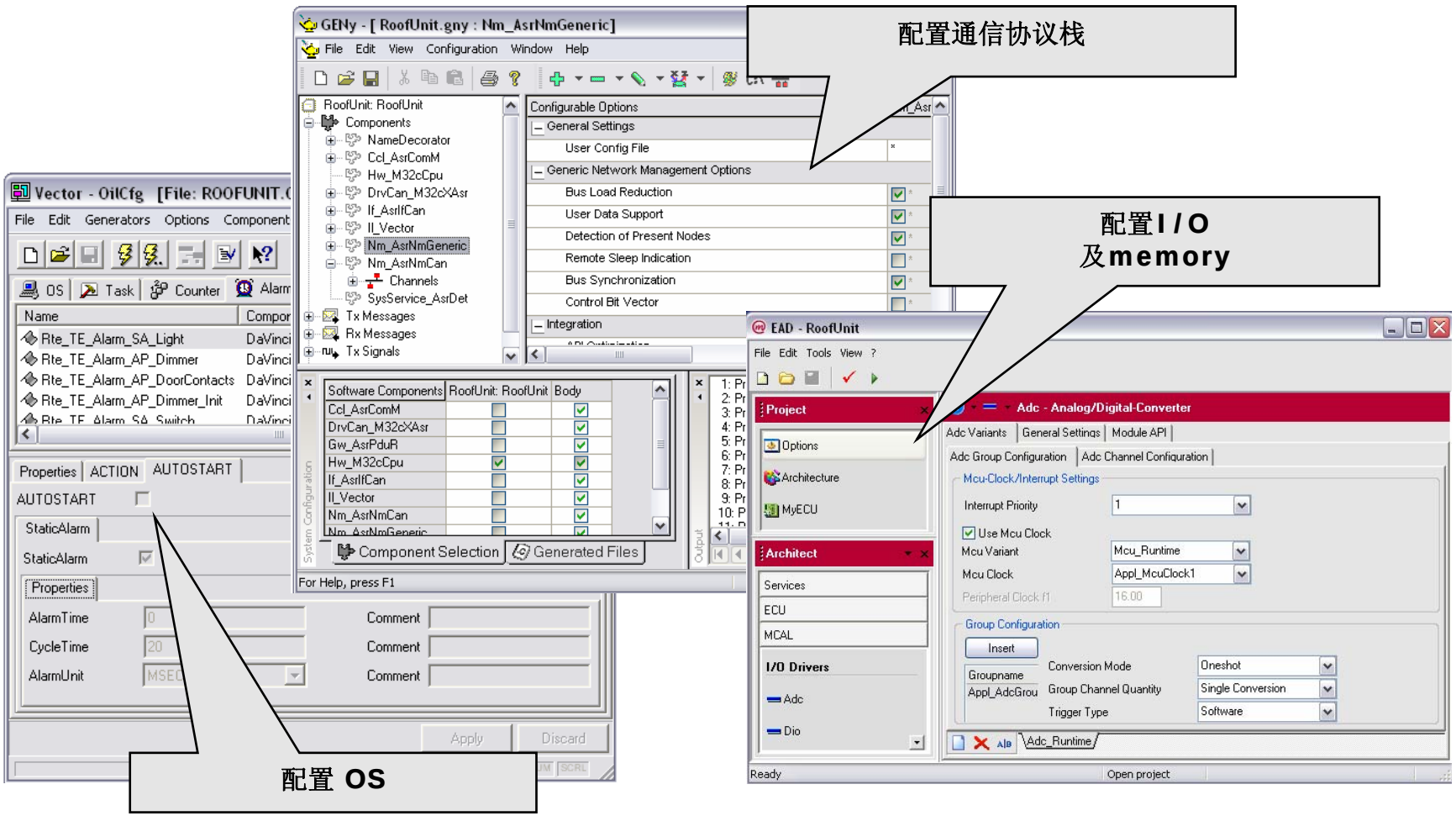




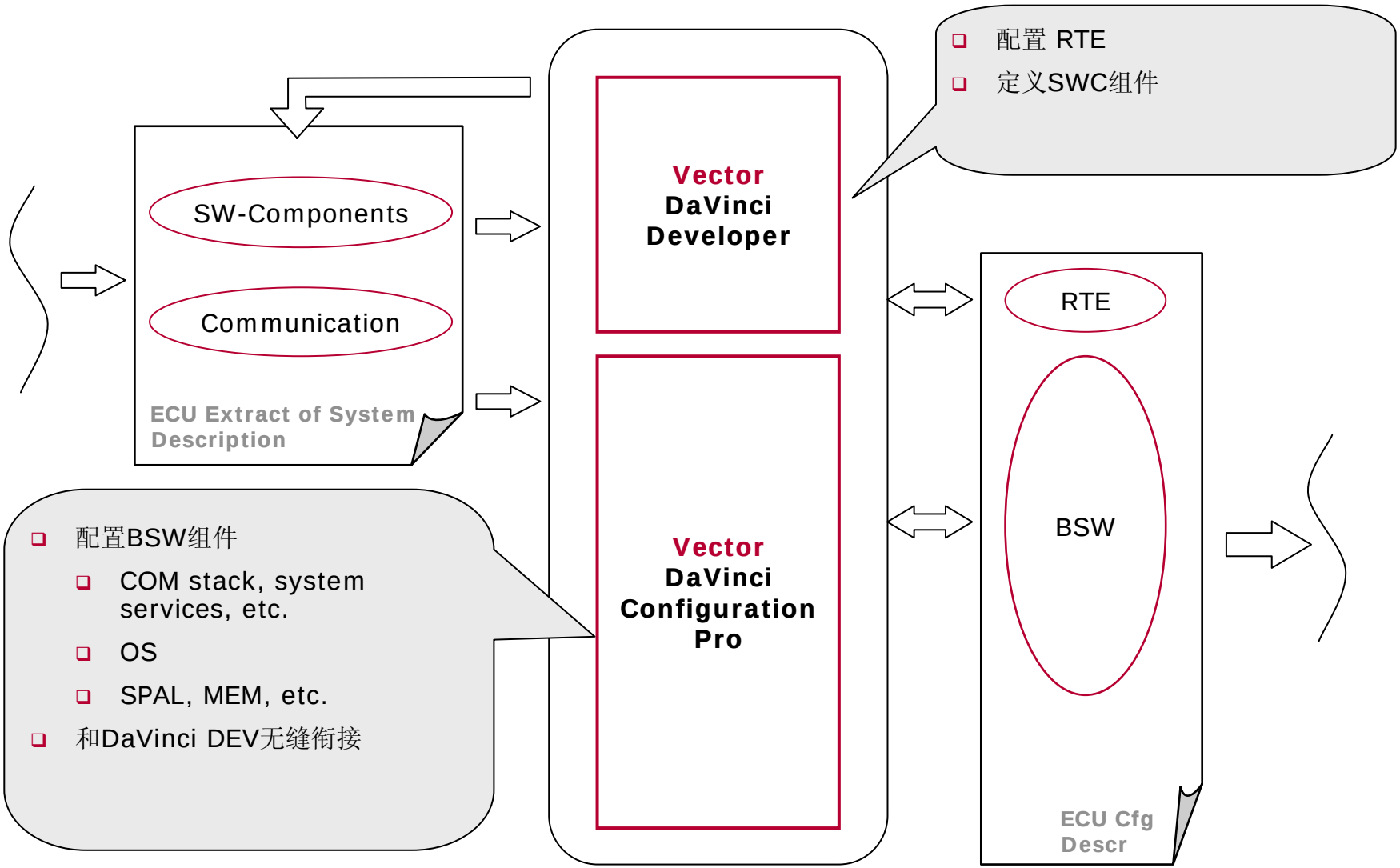


RTE 配置
(任务映射)

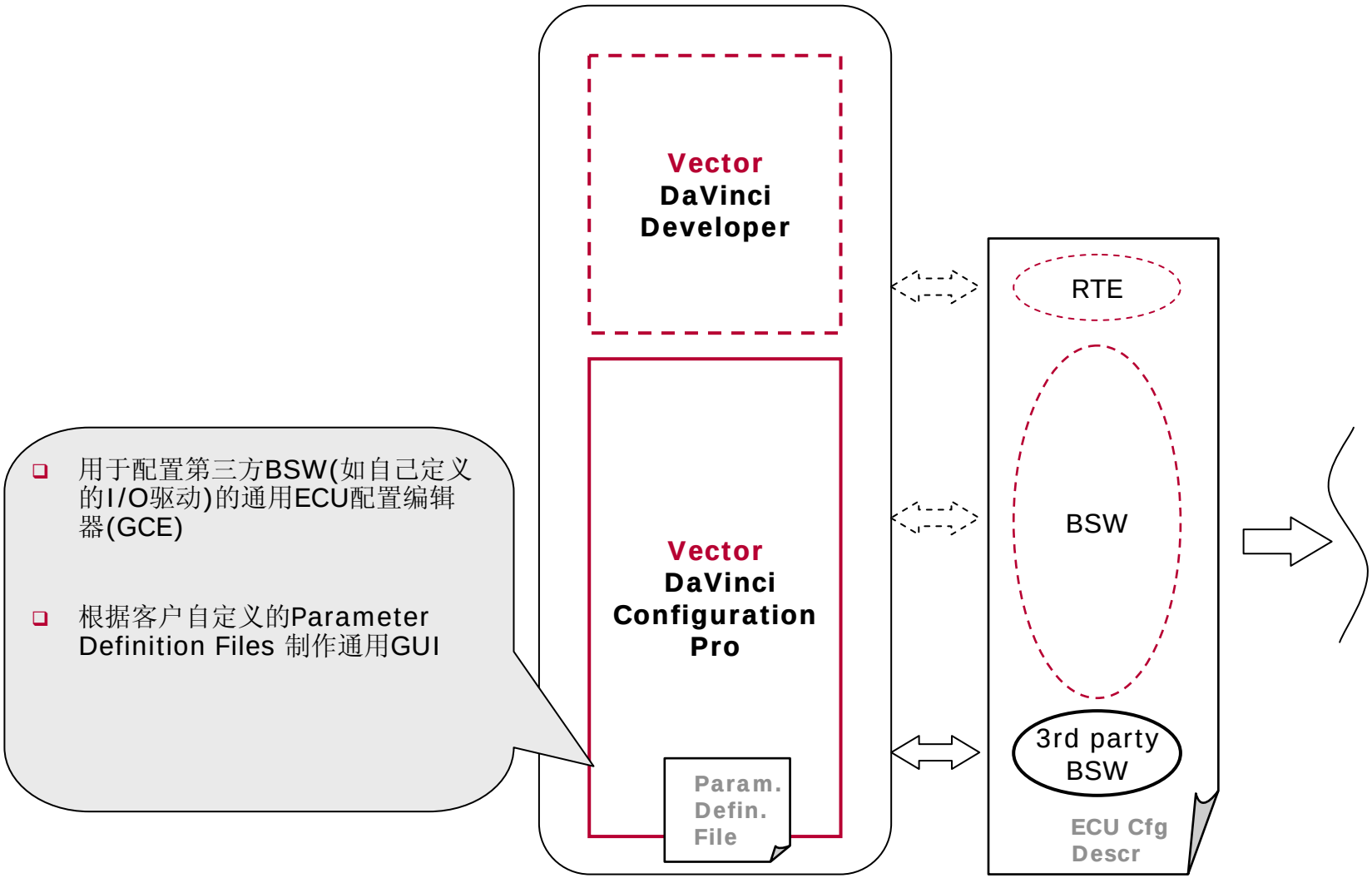
Tool: DaVinci Configurator Pro



ECU 配置

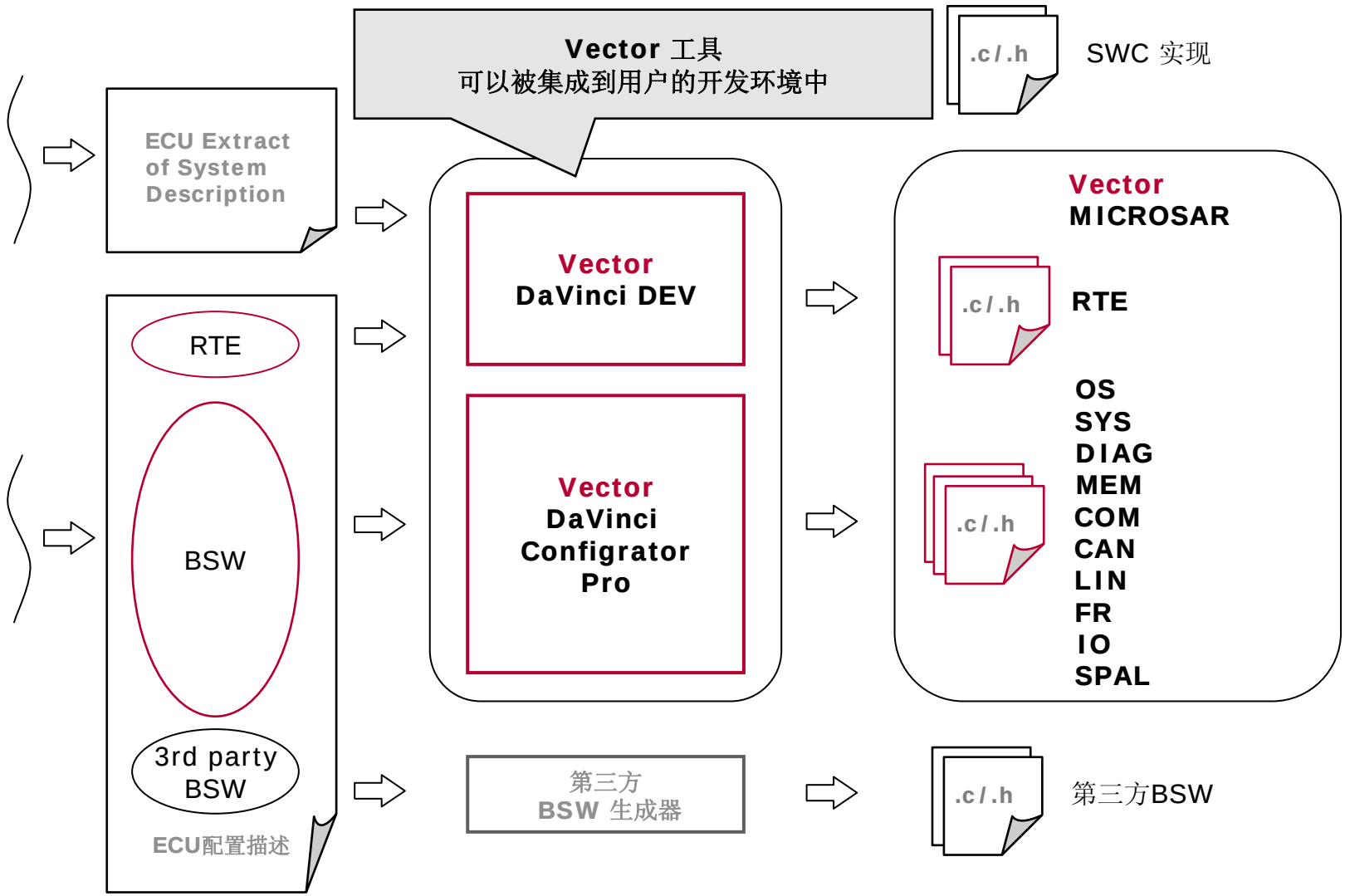


ECU 配置(第三方BSW)



- 用于配置第三方BSW(如自己定义的I/O驱动)的通用ECU配置编辑器(GCE)
- 根据客户自定义的Parameter Definition Files 制作通用GUI

ECU 代码生成



大纲

综述和目标

AUTOSAR入门 (introduction)

AUTOSAR方法论 (methodology)

AUTOSAR实时环境 (RTE)

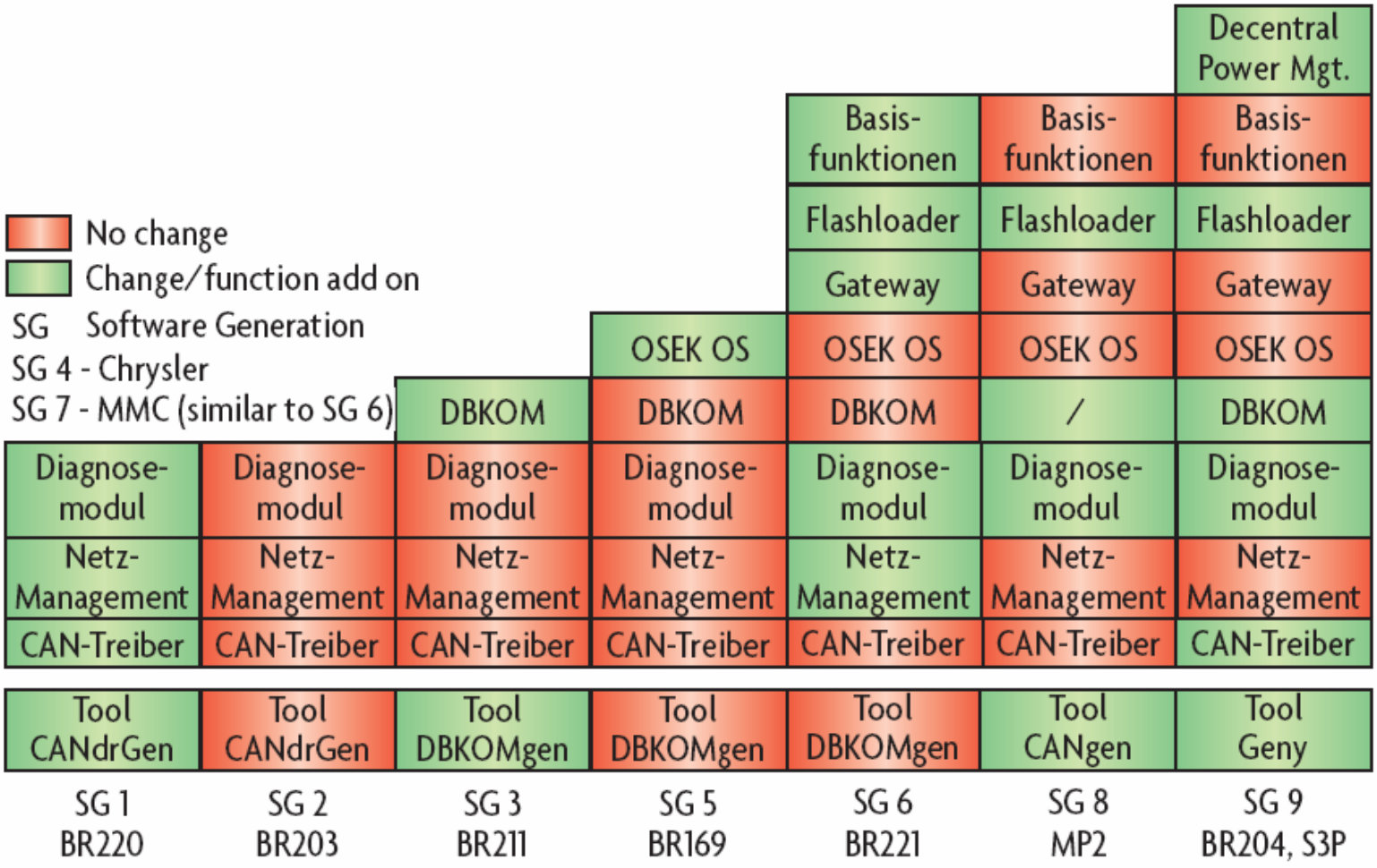
AUTOSAR基础软件 (BSW)

Vector AUTOSAR实现

> **从CANbedded到AUTOSAR**

蕴含和迁移

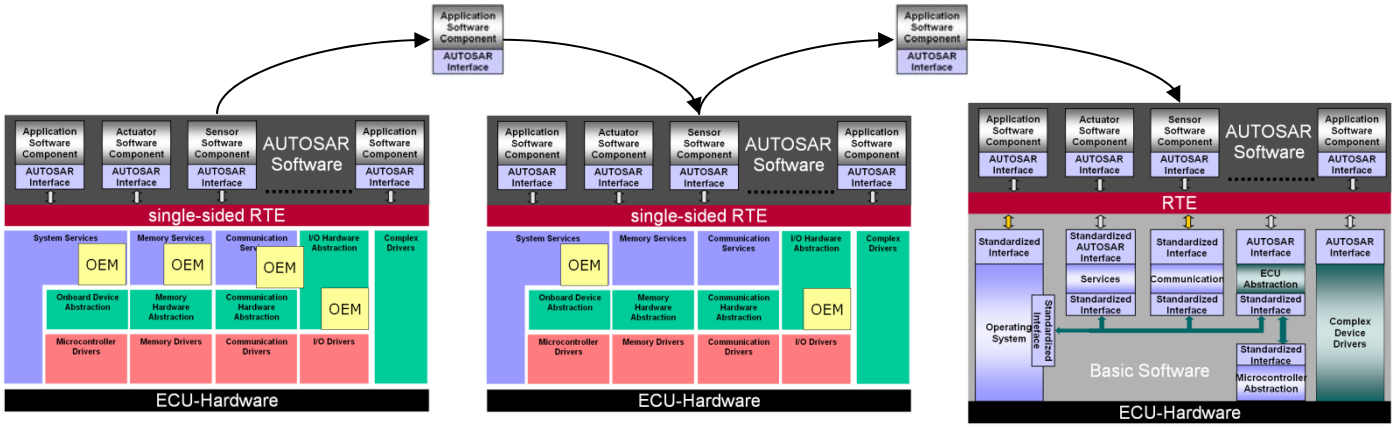
概述



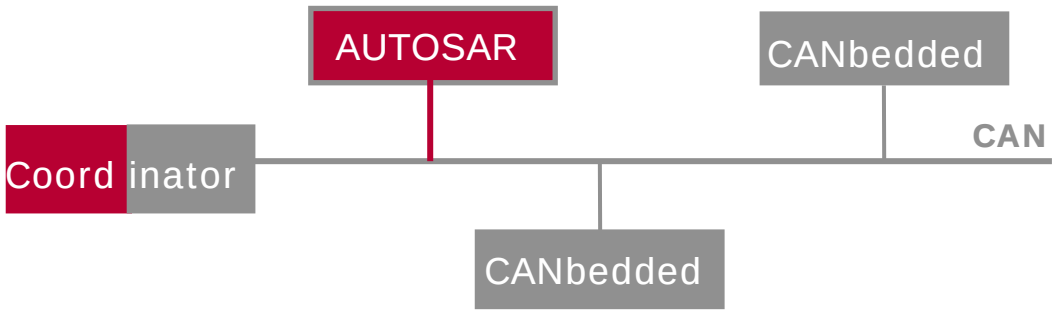
Source: Daimler AG

Overview

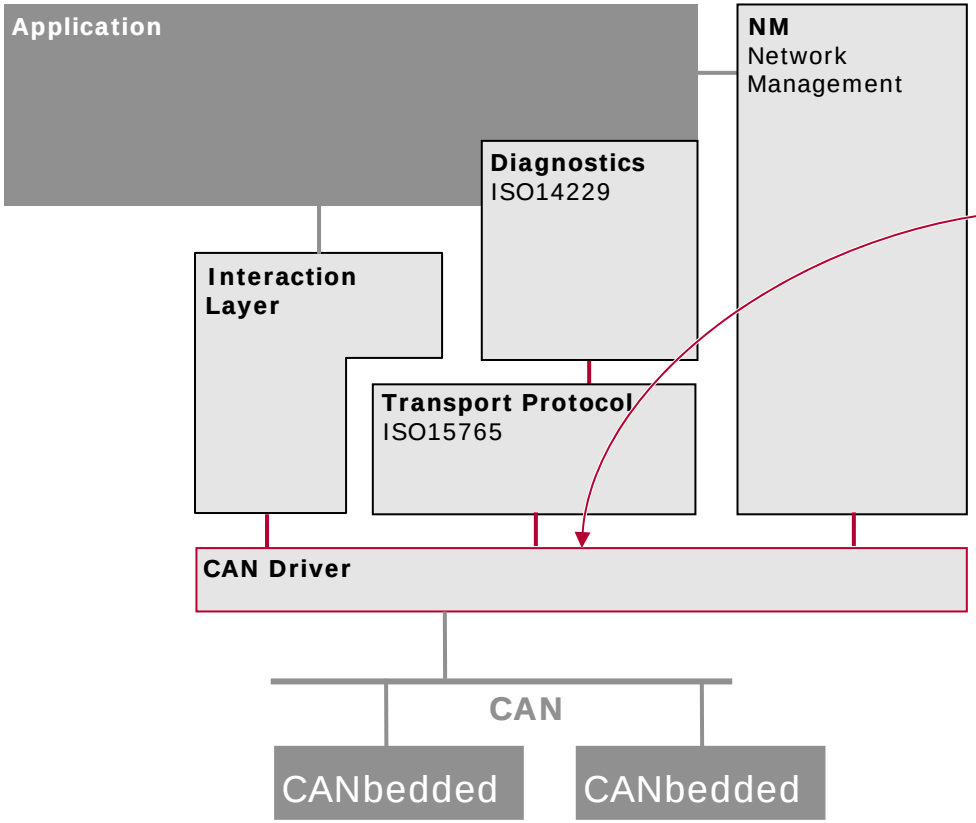
- ECU 中移植



- 网络中移植

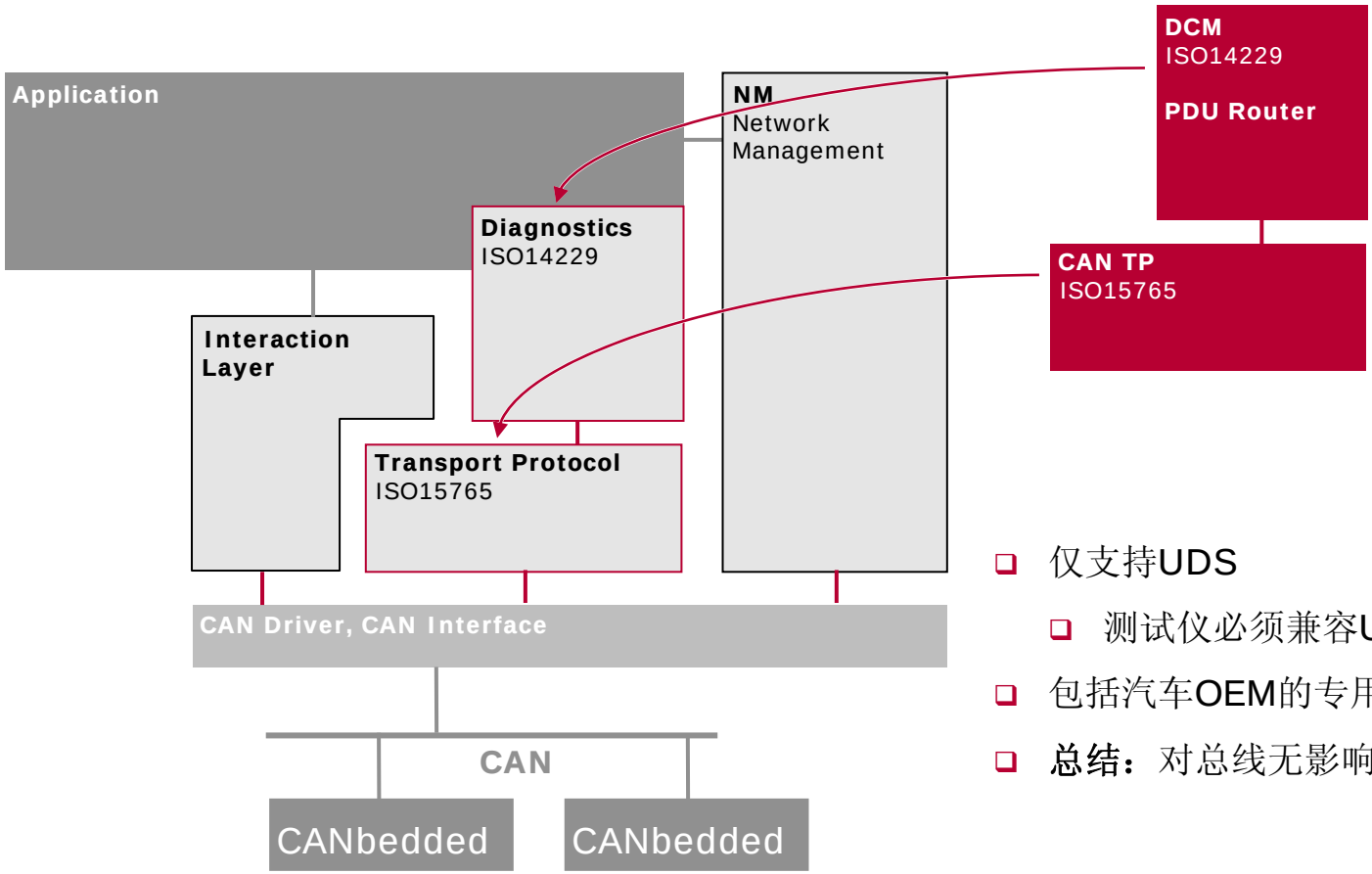


- AUTOSAR Driver



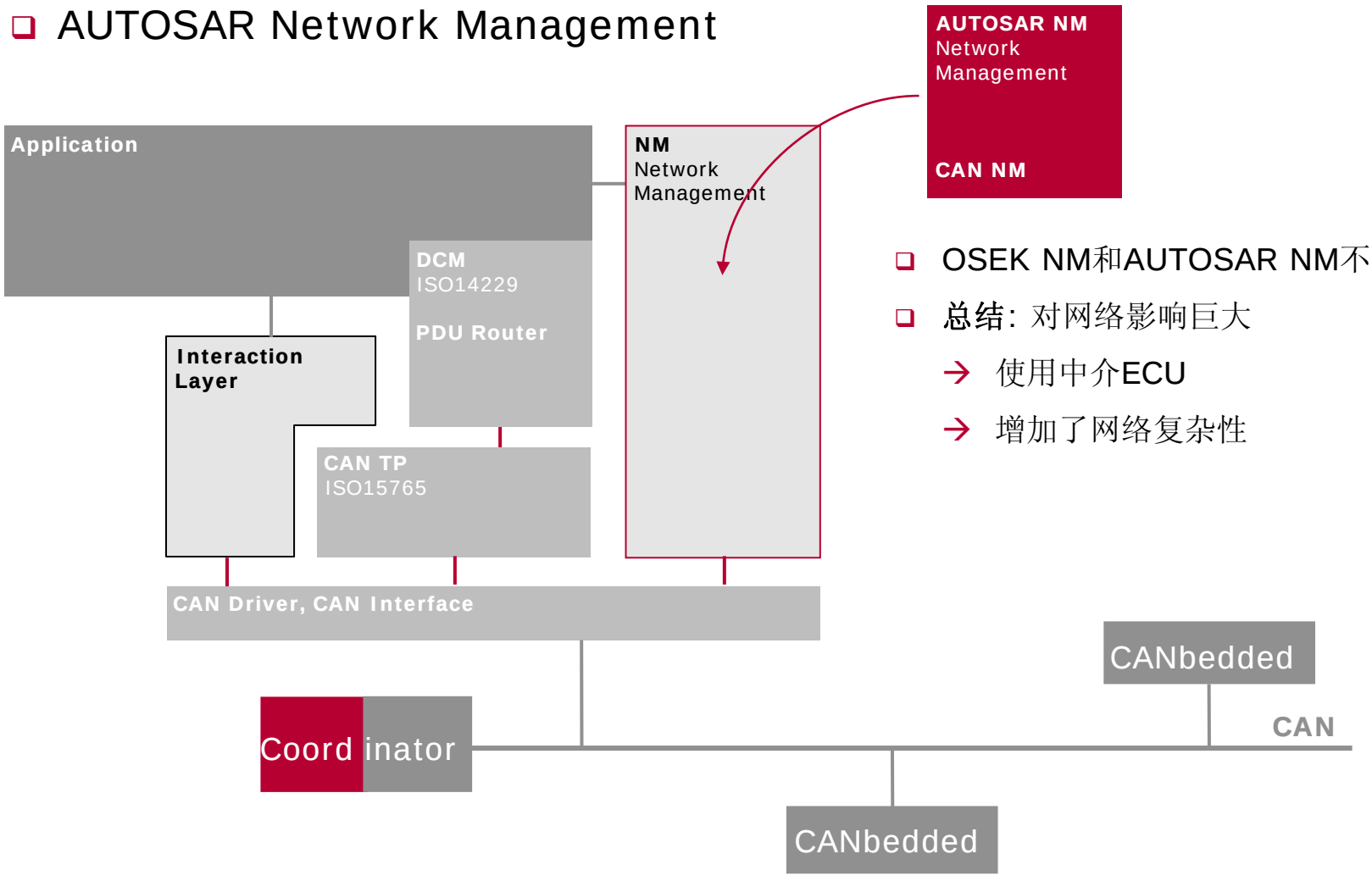
- 对总线没有影响
- 总线行为由CAN协议定义
- 对上层模块有影响
- 总结：对网络没有影响

- AUTOSAR Diagnostic and Transport Protocol



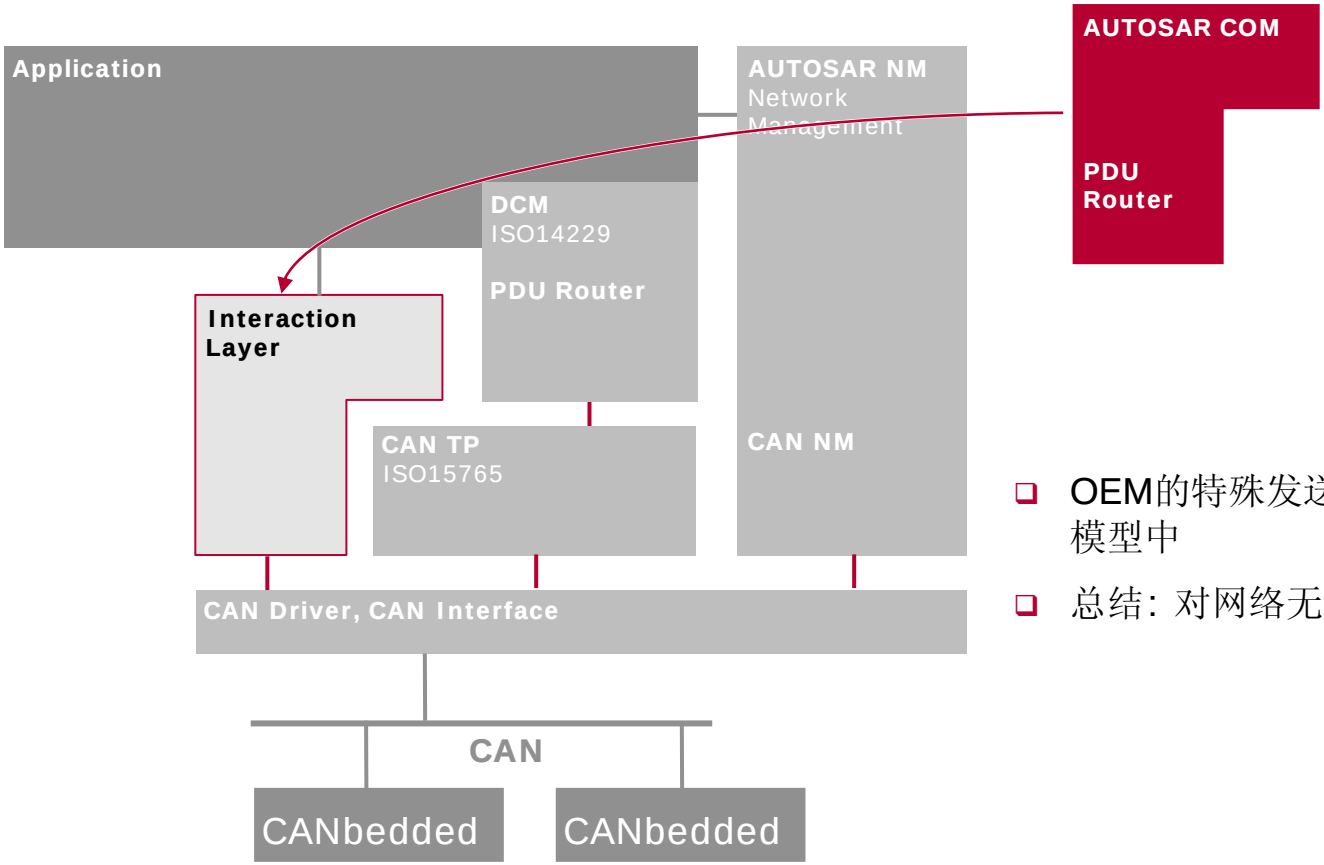
- 仅支持UDS
  - 测试仪必须兼容UDS和KWP2000
- 包括汽车OEM的专用诊断协议
- **总结：**对总线无影响，但对测试仪有影响

- AUTOSAR Network Management



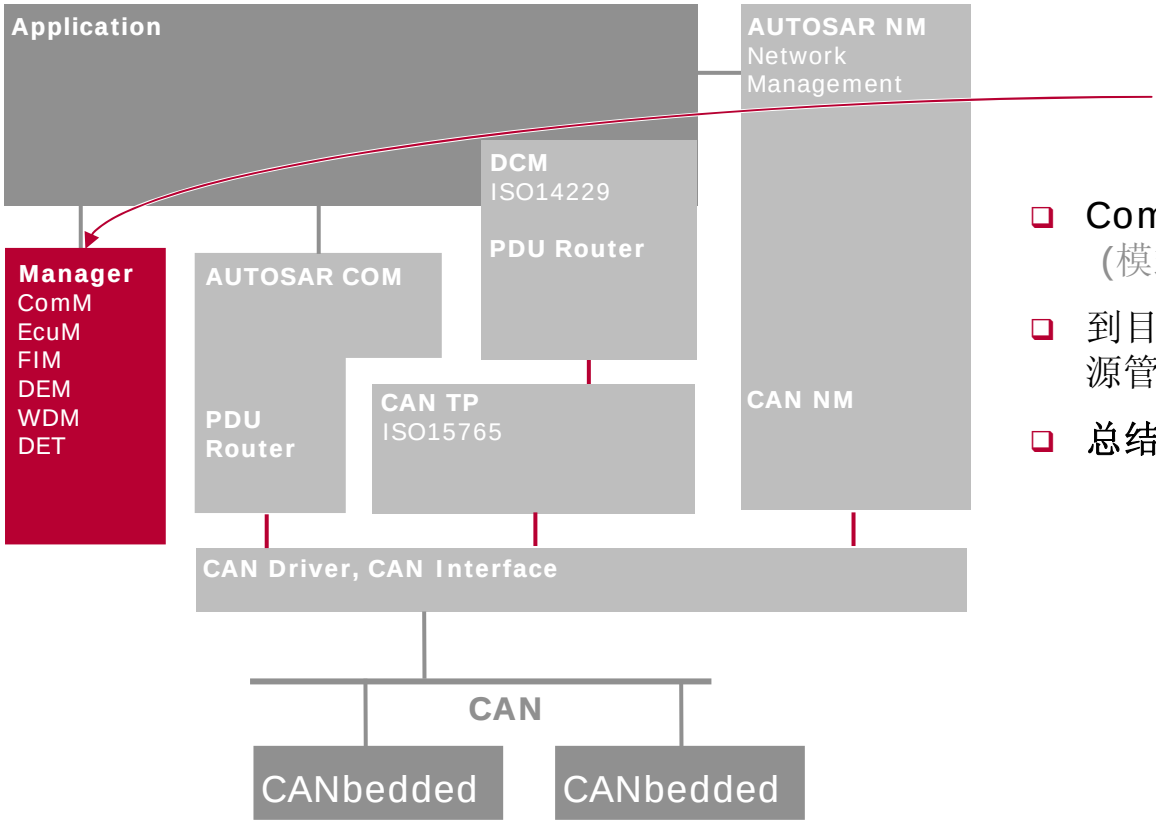
- OSEK NM和AUTOSAR NM不兼容
- 总结: 对网络影响巨大
  - → 使用中介ECU
  - → 增加了网络复杂性

AUTOSAR COM



- OEM的特殊发送方式并不在AUTOSAR模型中
- 总结：对网络无影响

AUTOSAR Manager Components

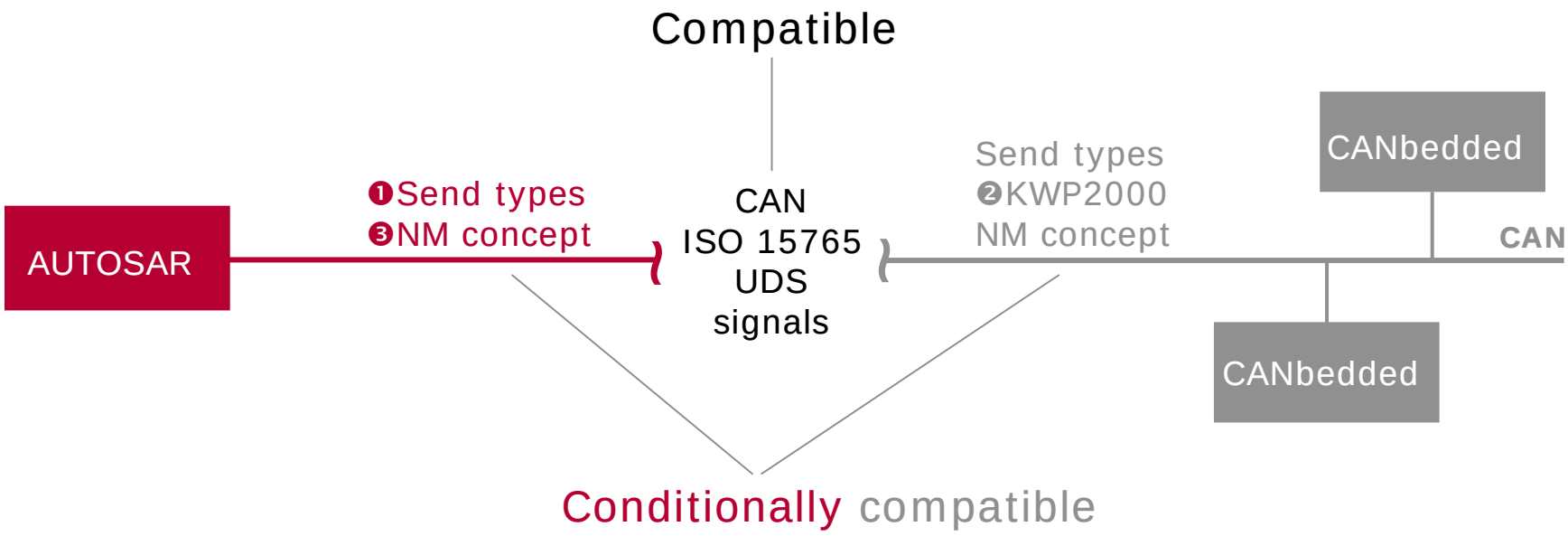


- ComM: 限制处理
  (模式管理)
- 到目前为止OEM专用状态管理(如，电源管理)
- **总结**: 对网络无影响

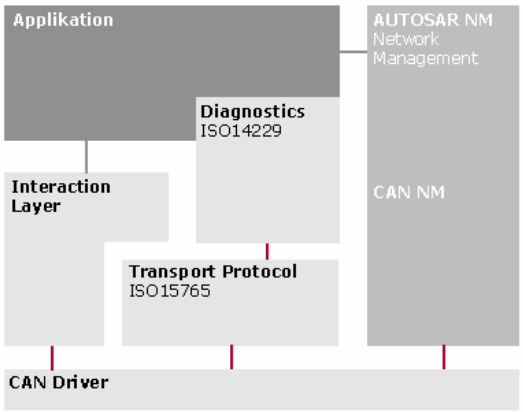
总结: Migration within a network

兼容性问题可能发生在:

- KWP2000 – UDS
- NM 概念
- 特殊发送方式



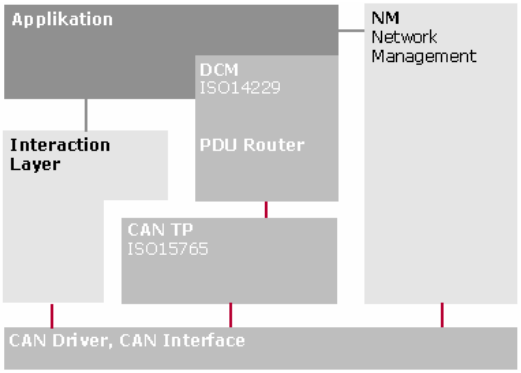
- 混合形式(OEM尝试的方式)



AUTOSAR NM

混合的策略：

- 尽可能少做改动，但是从网络上能兼容AUTOSAR



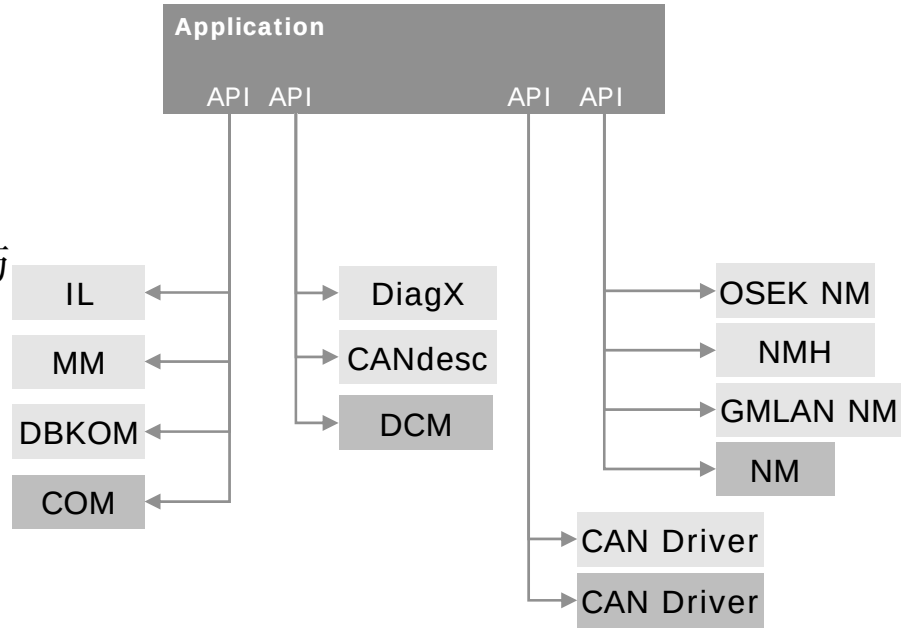
AUTOSAR driver, TP, diagnostics

- 保留节点在总线上的交互行为，兼容现有的ECU。

Migration within an ECU

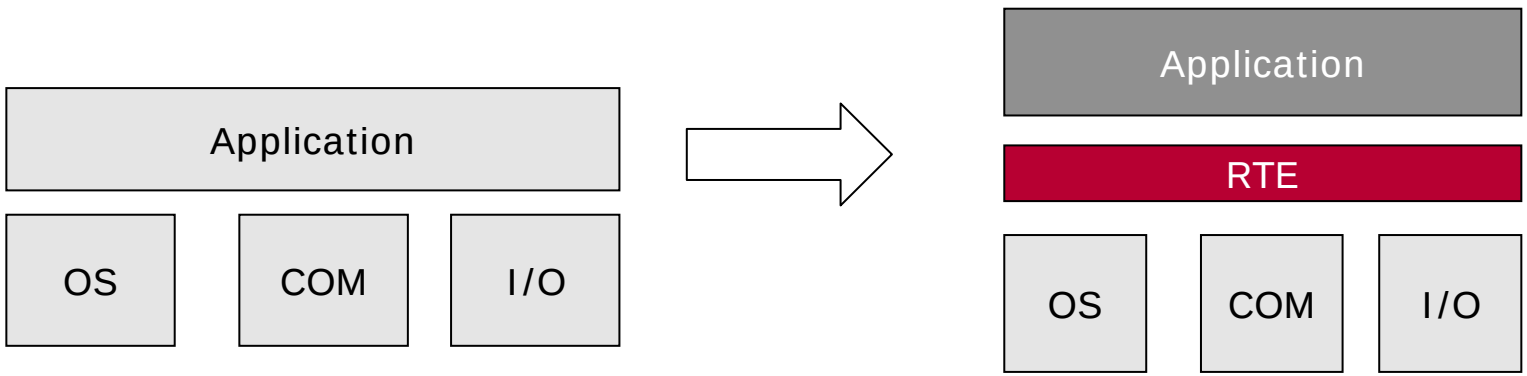
上述的混合模式对应用程序会造成什么影响？

- 目前应用程序已经需要很多种API
- 如果使用混合模式，将会有更多与AUTOSAR BSW之间的API出现
- 更多的软件版本



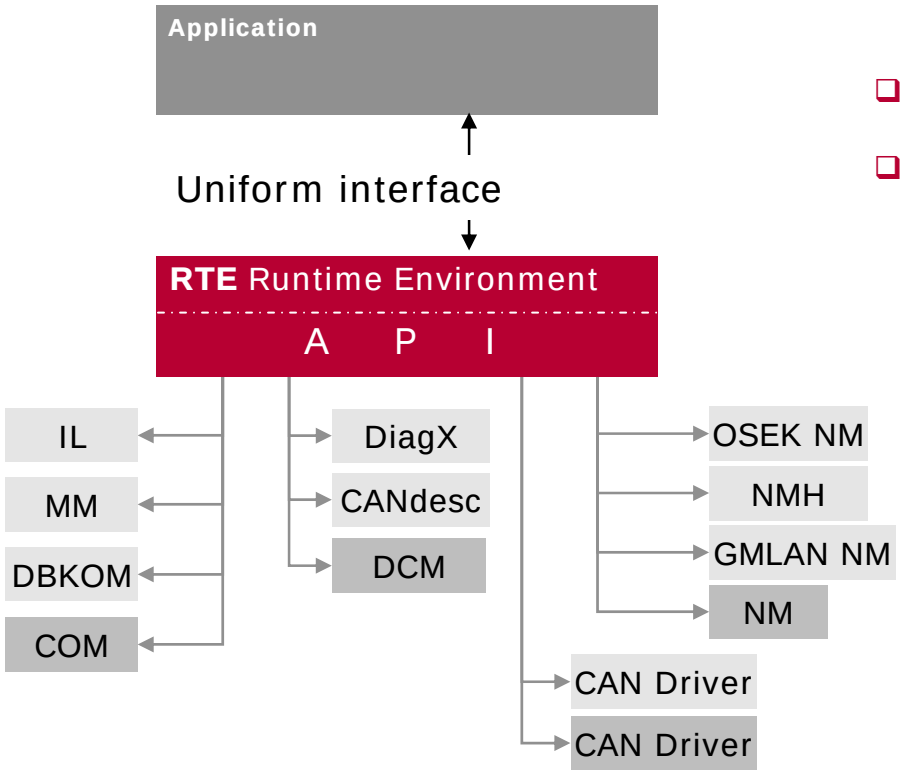
Migration within an ECU

- 架构分析
  - 类似架构
  - 应用程序与BSW交互
- 解决方案策略
  - AUTOSAR：应用程序之和RTE交互
    - 标准化应用程序或SWC的接口
  - 总结：移植到AUTOSAR需要在应用程序和BSW中间加入RTE



Migration within an ECU

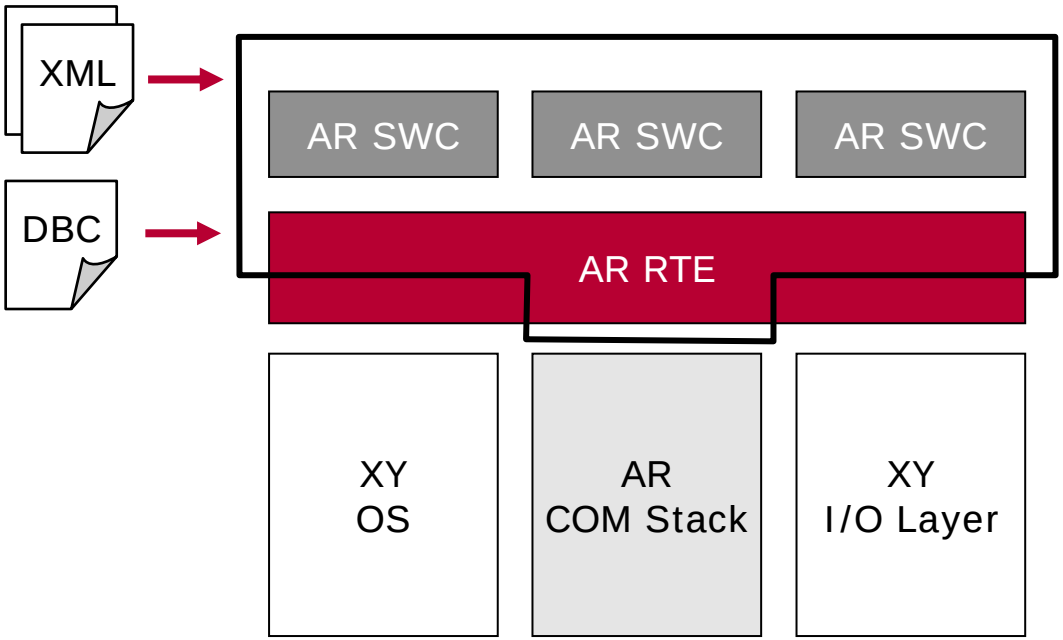
方法1:　单边 **RTE**



- 仅仅多了一个接口：RTE
- 使得应用程序和下层BSW无关

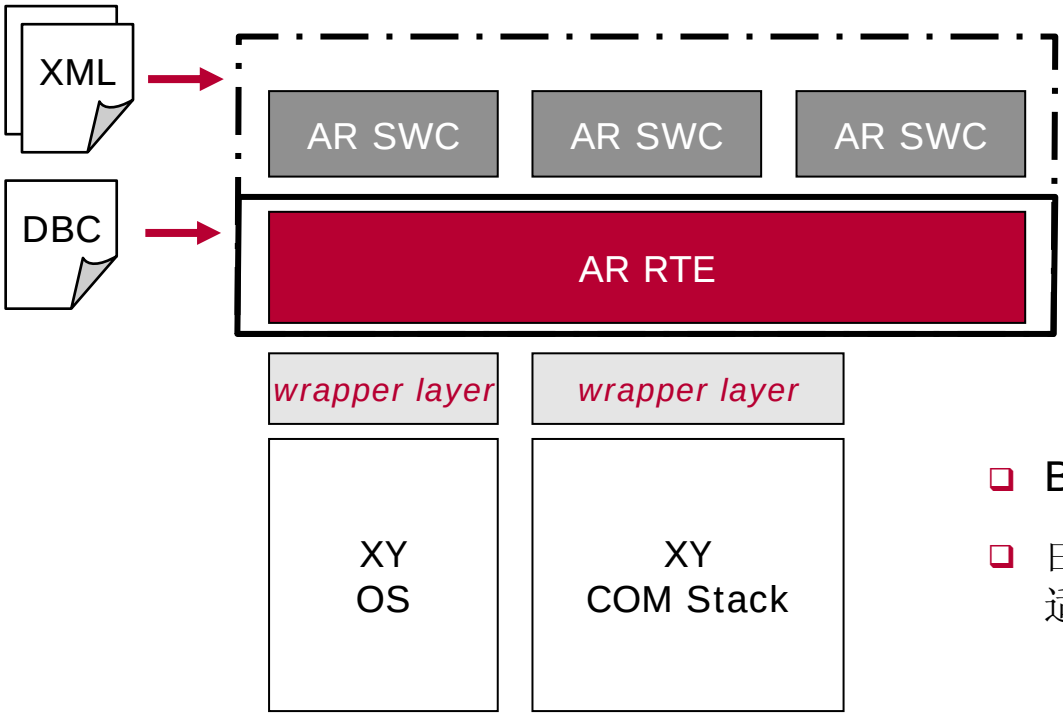
ECU中的移植

方法2： 单 / 双边 RTE



- 使用AUTOSAR工具来:
  - 导入AUTOSAR SWC的XML
  - 也可以建立新的SWC
  - 配置RTE
  - 生成RTE
- 完成RTE-API的以下接口:
  - XY OSEK API
  - XY I/O Layer

- 方法3：双边 **RTE**



- BSW中的接口层用于转换RTE-API
- 由于BSW的接口由接口层完成，所以适用于任何RTE生成器

移植总结

- RTE提供了什么?
  - 完全实现建模开发方法
  - 供应商可以移植到AUTOSAR系统，而不限制于汽车OEM
  - 对于任何移植步骤而言，应用程序都是独立的

- 总结
  - AUTOSAR BSW的分布与当今的软件策略吻合，这使得移植变得可行。
  - AUTOSAR提供了“big unknown”式的RTE，允许在不需要大量额外工作的情况下，将系统移植到AUTOSAR规范下

AUTOSAR对合作者们的优势

- 汽车 OEMs
  - 使OEM具有跨平台的汽车功能
  - 使OEM注意力能集中于与竞争力相关的软件开发
  - 允许汽车的型号和配置进一步的多样化
  - 使软件认证有标准可依
  - OEM之间能交换他们的创新功能
- 供应商
  - 使供应商能共享开发过程
  - 使五花八门的OEM规范变得一致
  - 提高应用功能相关软件的开发效率
  - 允许新的商业模式建立

AUTOSAR对合作者们的优势

- 工具供应商
  - 通过标准接口，使开发过程互相兼容
  - 使五花八门的客户需求变得一致
  - 工具链能够更容易的被第三方工具所填补
  - 更简单的集成到已有的工具链中
  - 支持从设计到代码生成的整个过程
- 市场中的新角色
  - 成熟的标准降低了进入市场的门槛
  - 有助于发展新型的商业模式