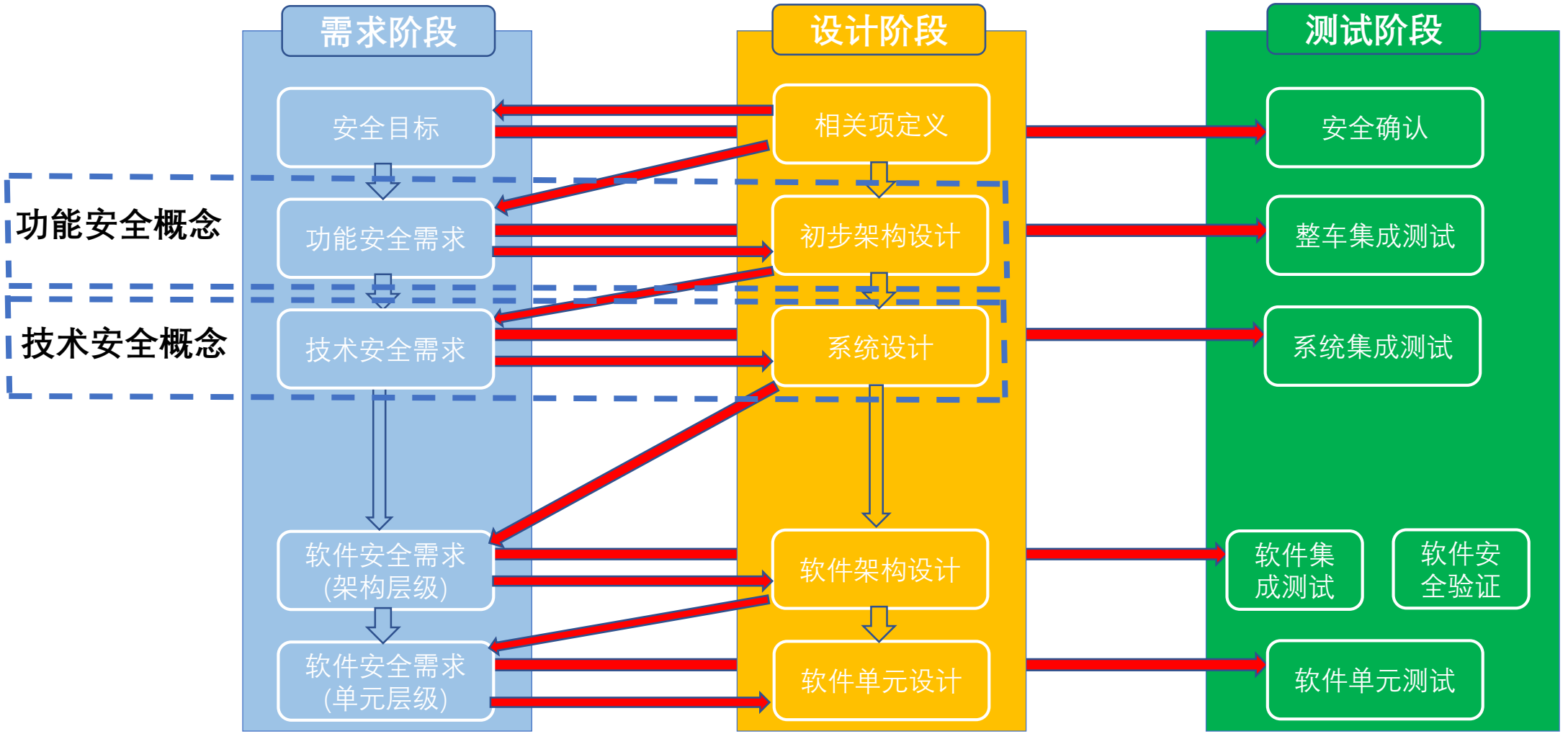
momenta

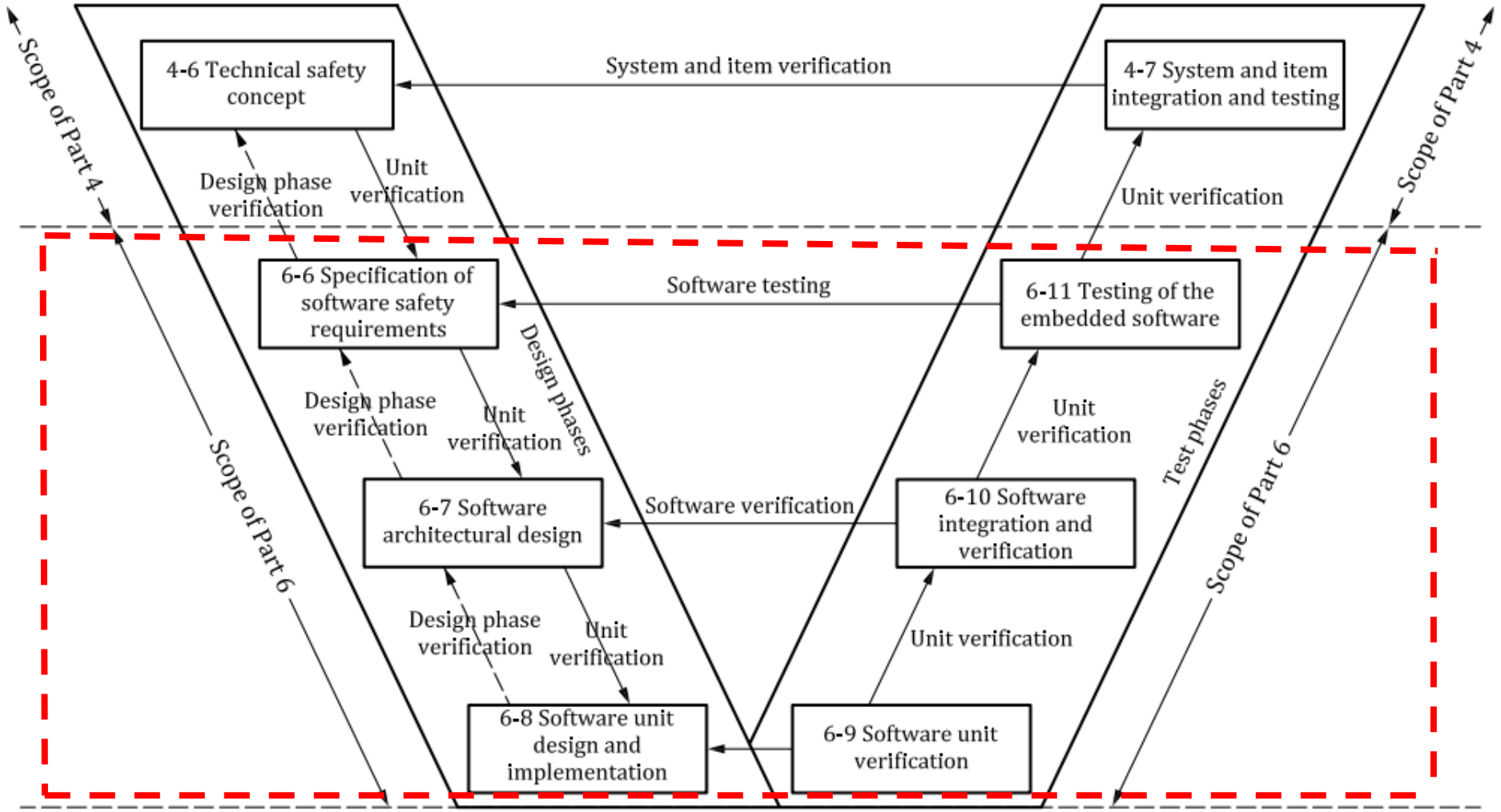
Workshop
软件功能安全开发介绍

2021/07

功能安全概念到软件开发流程

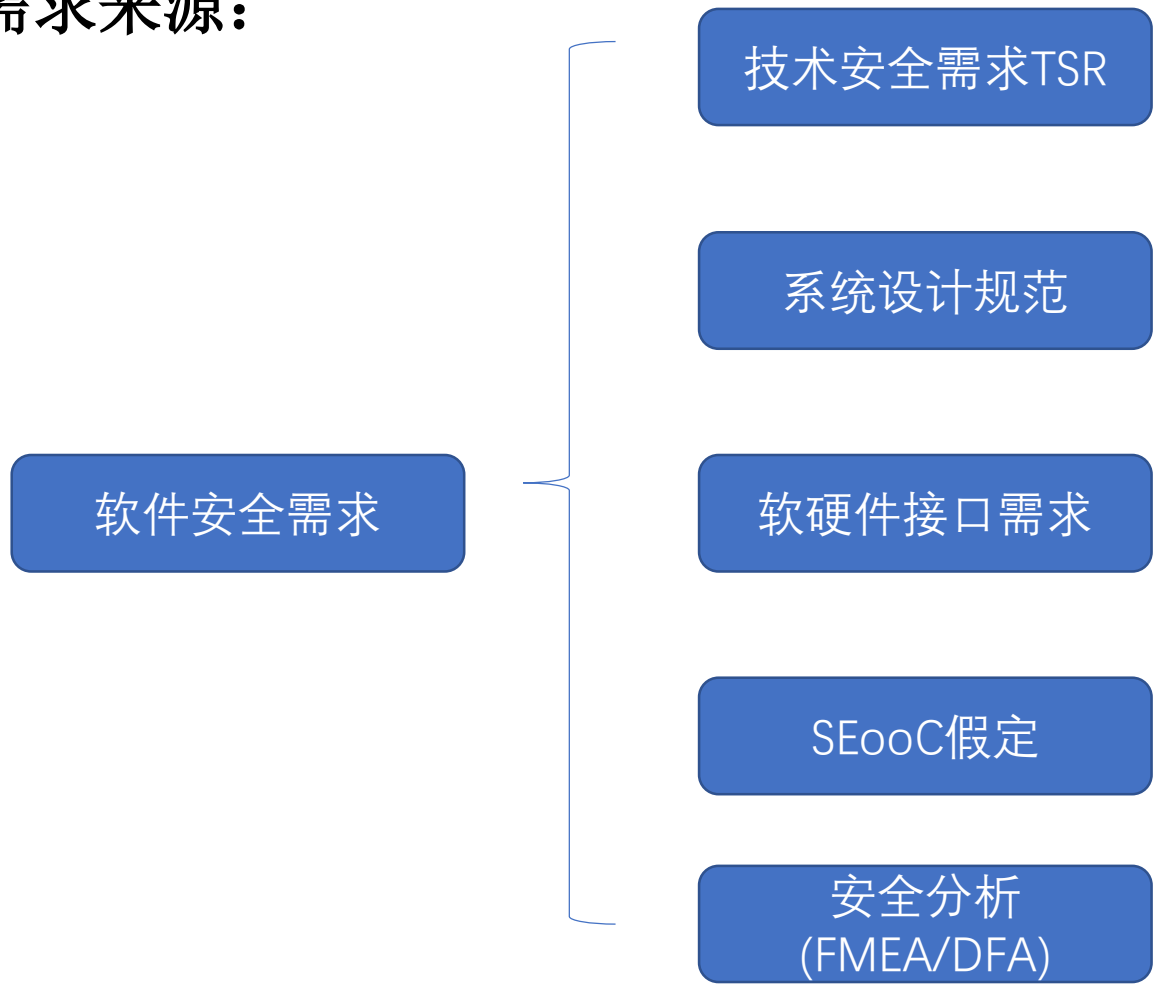


软件开发流程



软件安全需求

软件安全需求来源：



软件安全需求

软件需求规范：

- 软件安全功能和特性相关的安全需求
- 软件安全需求规范应包含：
  - ✓ 系统和硬件配置
  - ✓ 软硬件接口规范
  - ✓ 时间约束
  - ✓ 外部接口
  - ✓ 工作模式和模式跳转
- 软硬件接口相关的安全需求
- 软件需求的ASIL分解(若适用)
- 软件安全需求验证：合适性, 符合性, 一致性

软件需求特性：

- 无歧义并可理解
- 不可分割
- 内部一致
- 可实现的
- 可验证的
- 完整性

软件架构设计

软件架构设计规范：

- 软件架构设计应满足功能安全需求且可验证
- 遵循相应的设计准则，以避免系统性故障
- 软件架构设计应包括静态设计和动态设计
- 软件架构安全分析(FMEA，DFA)
- 故障检测和处理安全措施/机制
- 软件资源评估(执行时间，Memory空间，通讯资源)

软件架构设计

软件架构设计原则：

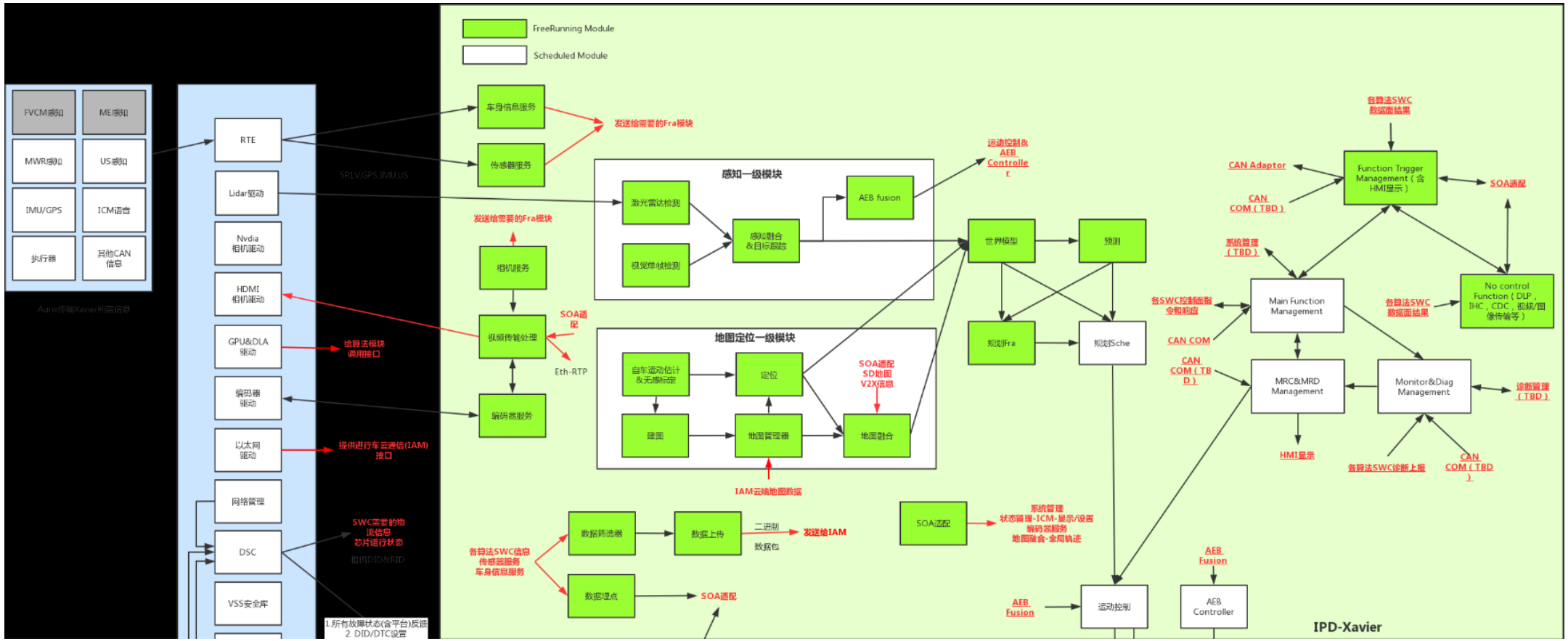
- 采用合适的层级架构
- 限制软件组件的大小和复杂度
- 限制接口大小
- 组件内部的强内聚力
- 软件组件间的低耦合度
- 合适的调度属性
- 限制使用中断
- 采用合适的空间隔离

软件架构特性：

- 可理解性
- 一致性
- 简单性
- 可验证性
- 模块化
- 抽象性
- 封装性
- 可维护性

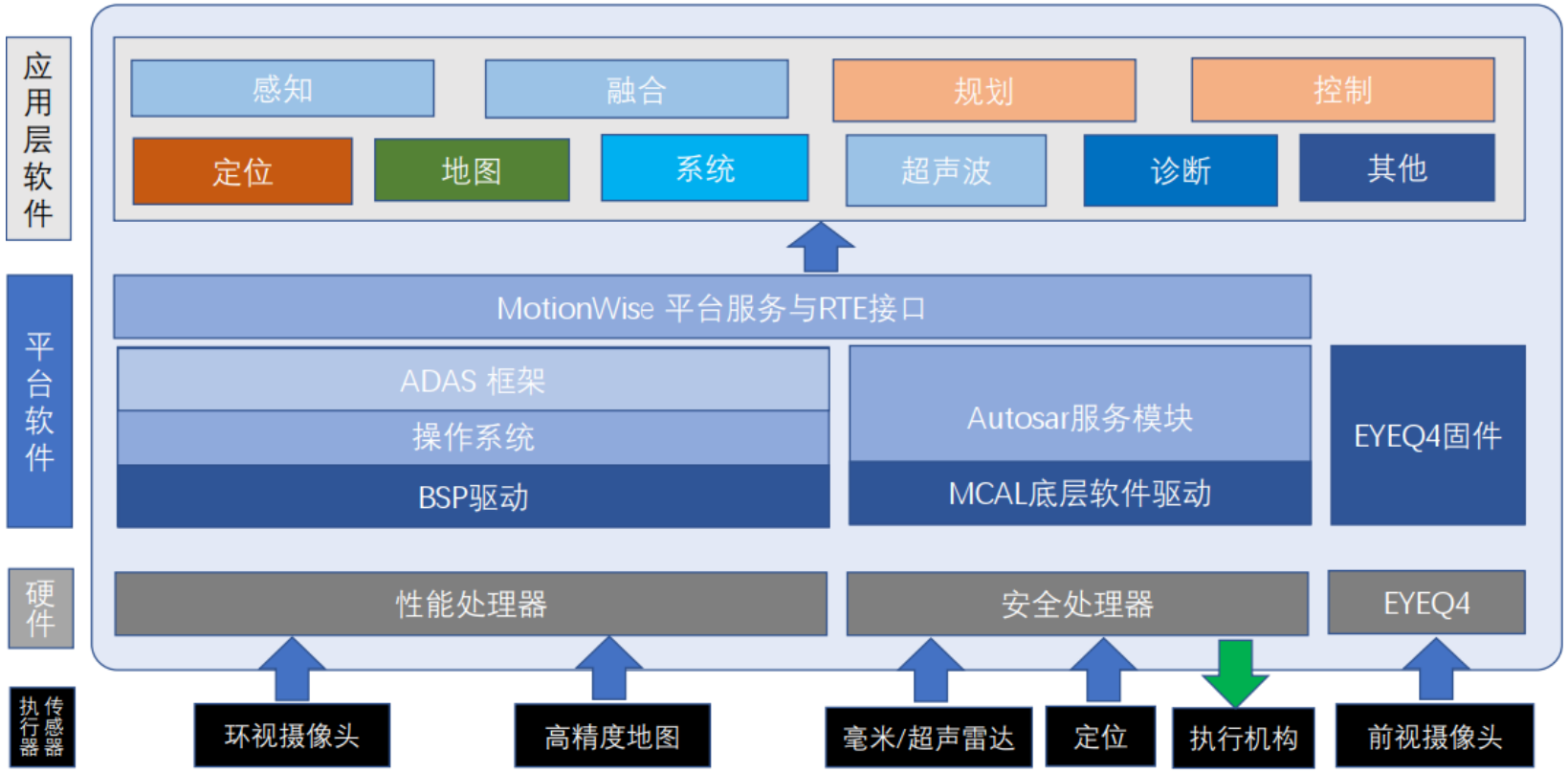
软件架构静态视图

包含所有的模块和接口，以及模块之间的数据流。

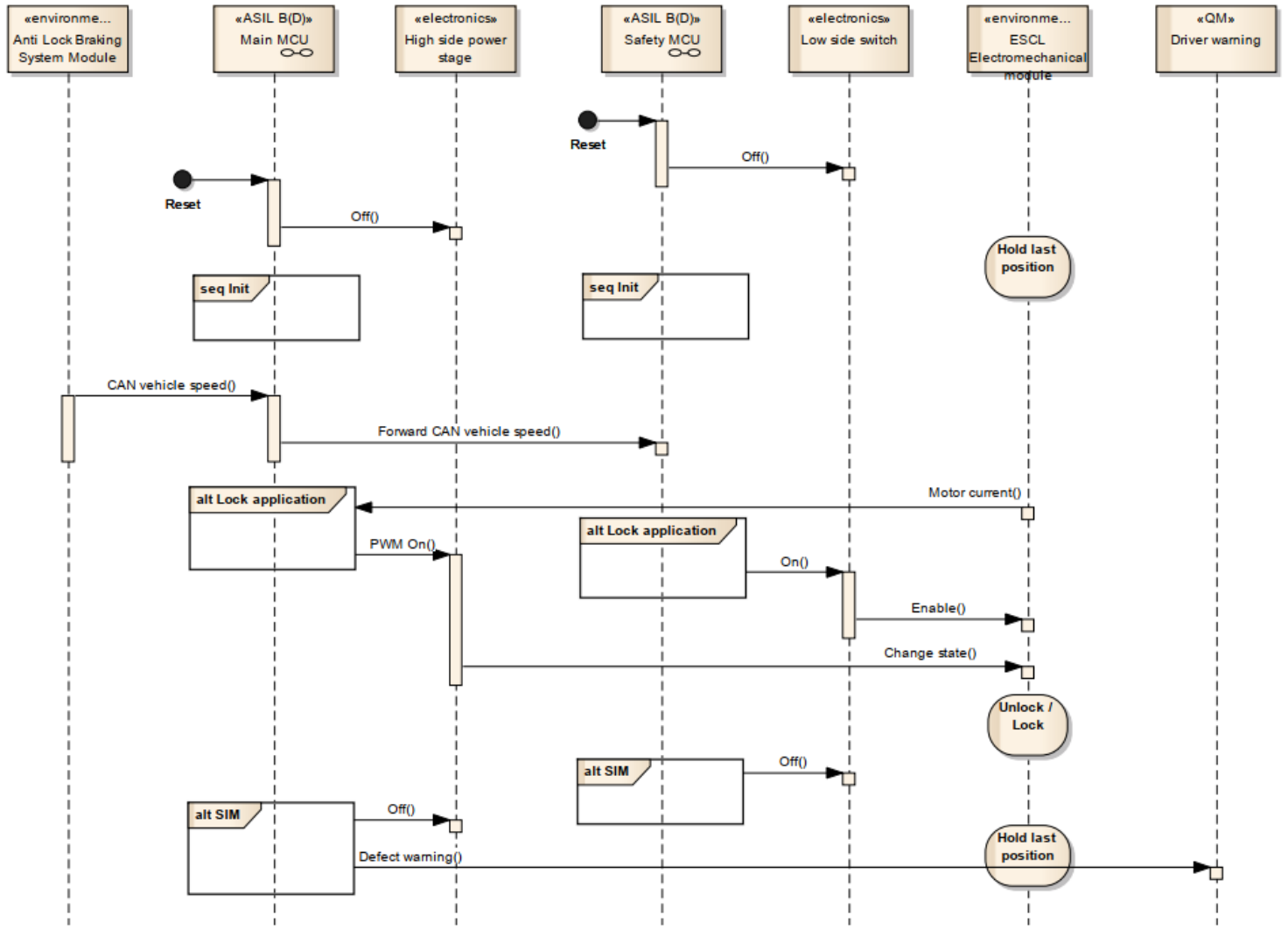


软件架构分层

显示软件的层级划分，以及各软件模块所在的层级。



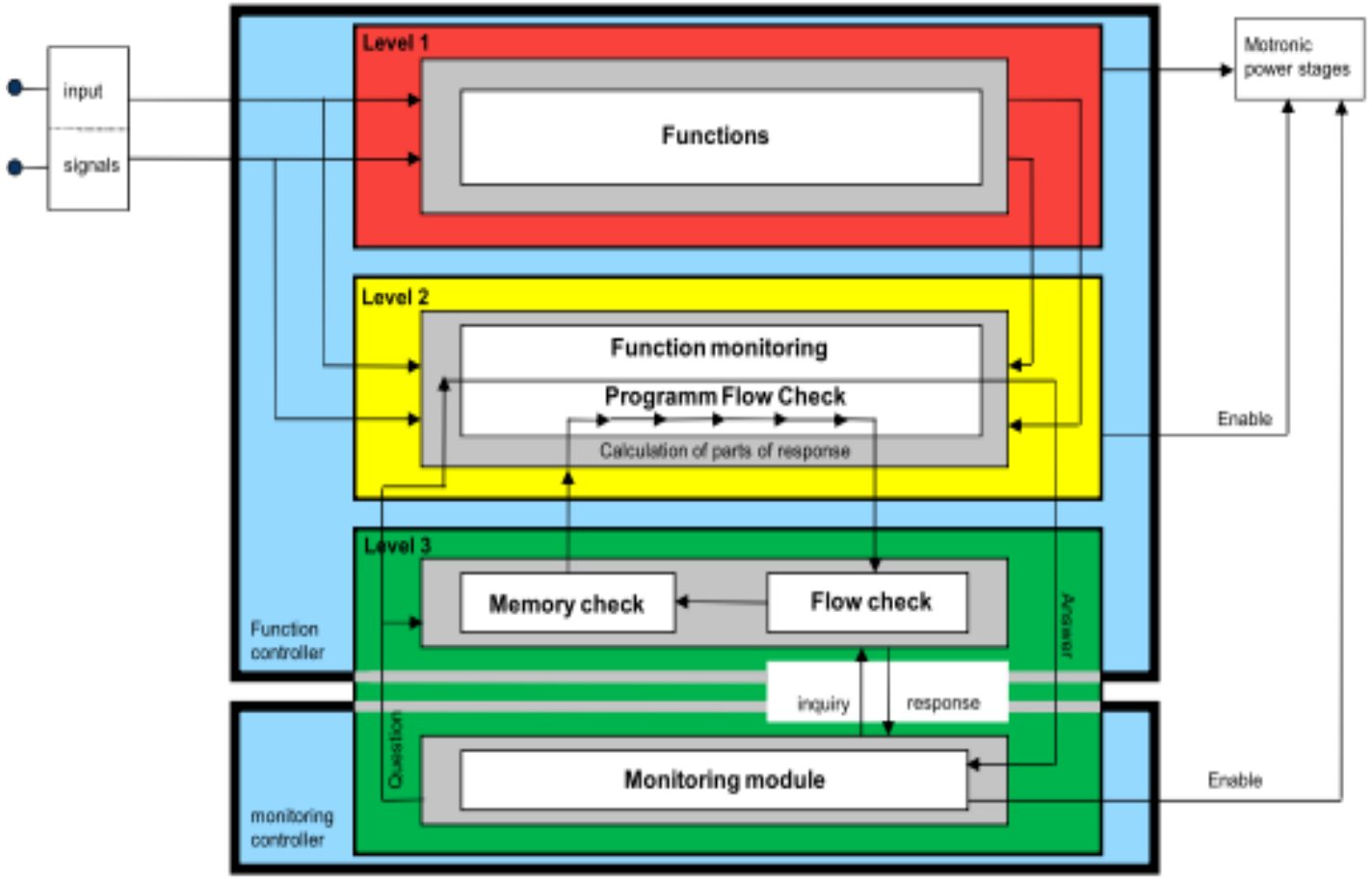
软件架构动态视图



软件安全话题

- ✓ 系统诊断功能（主要在App层）
  - 传感器多路冗余校验
  - 执行器反馈
  - 通讯故障
  - ...
- ✓ 硬件诊断功能（主要在MCAL层）
  - RAM/ROM/FLASH
  - CPU
  - Register
  - ...
- ✓ 软件自身失效

软件安全架构(E-GAS)



特点：

- 硬件—双芯片架构
  - 功能芯片
  - 监控芯片
- 软件—三层结构
  - Level 1：应用功能层
  - Level 2 ：功能监控层
  - Level 3 ：控制器监控层

软件架构设计验证

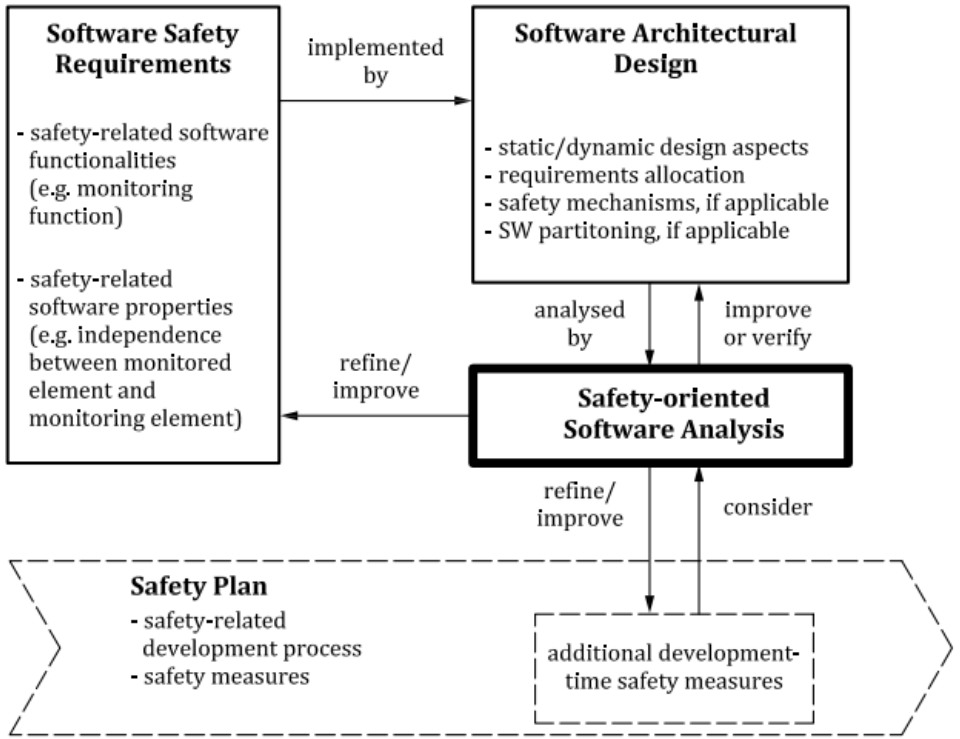
软件架构设计验证方法：

- 走查
- 检查
- 动态行为仿真
- 原型产生
- 正式验证
- 控制流分析
- 数据流分析
- 调度分析

安全分析

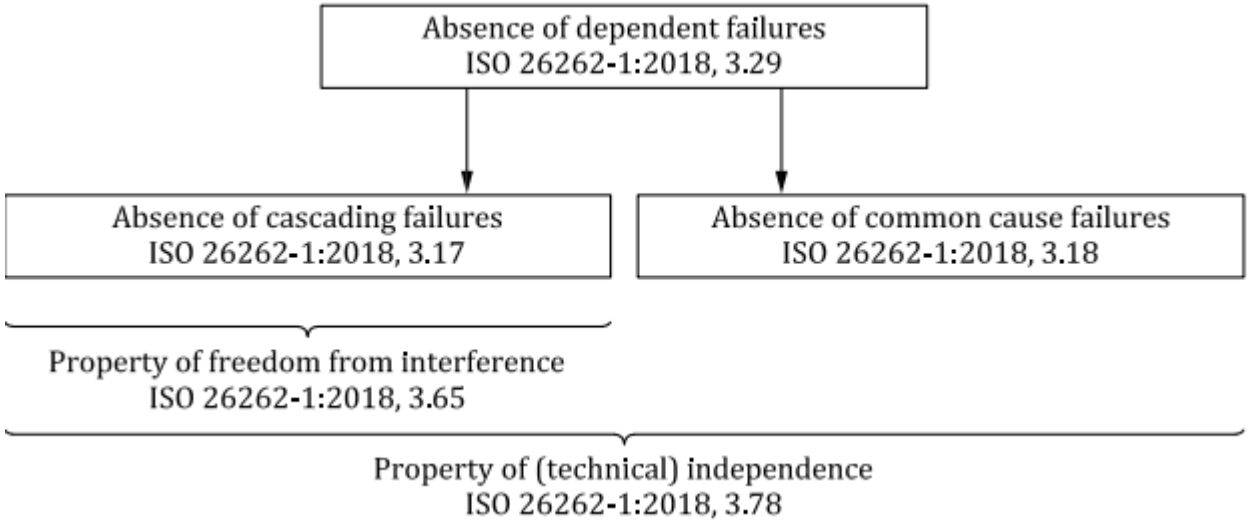
安全分析用以确认软件能提供ASIL所要求的规定的功能，行为和完好性能力。

常用的软件安全分析方法有FEMA和FTA。

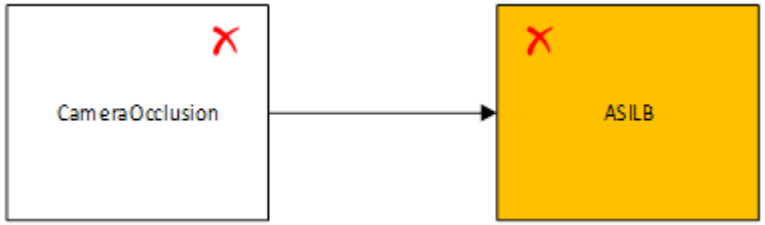


相关失效分析DFA

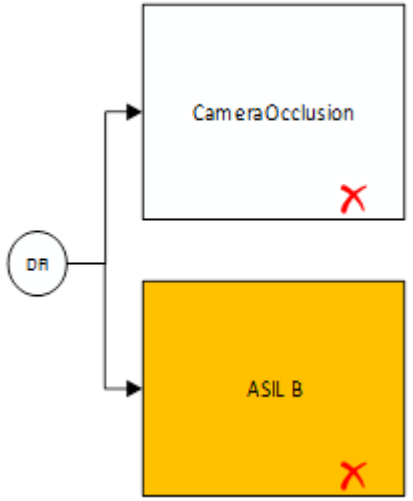
关联分析主要考虑系统/软件元素可能存在的共因失效和级联失效，并采取安全措施以减轻关联失效，以满足独立性要求。



级联失效



共因失效



软件单元设计

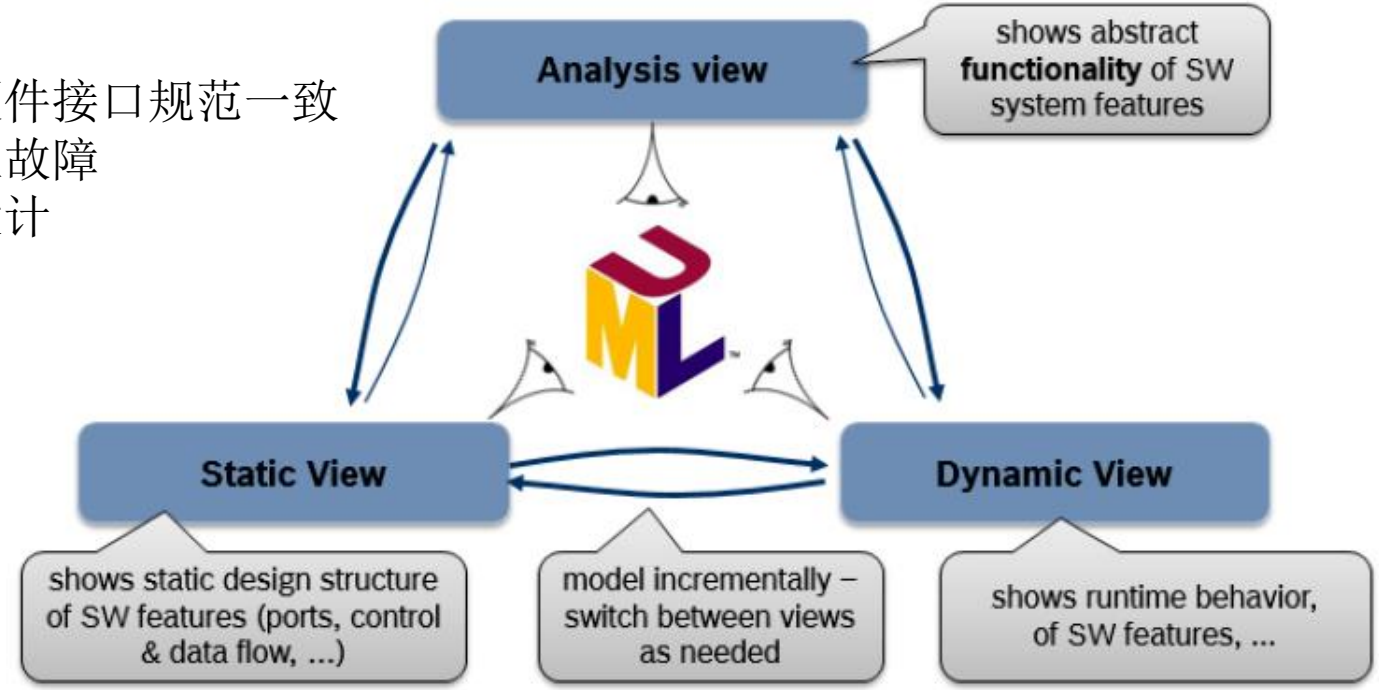
目的

- 根据架构设计，设计准则和软件安全需求，开发软件单元设计
- 执行软件代码开发

软件单元设计规范包含如下内容：

- 软件单元设计要满足相应的软件安全需求
- 软件单元设计应与软件架构设计规范和软硬件接口规范一致
- 软件单元设计应遵循相关准则，避免系统性故障
- 软件单元规范应描述单元功能行为，内部设计
- 软件单元设计和代码实现应遵顼原则：
  - ✓ 接口一致性
  - ✓ 正确的数据流和控制流
  - ✓ 简单，可读，易于理解
  - ✓ 鲁棒性
  - ✓ 可验证性

UML软件单元设计如右图所示：



软件校验与测试

➢ 单元验证和测试

验证方法

✓Walk-through/Inspection
✓静态代码分析
✓数据流和控制流分析
✓故障注入测试
…

测试用例选取方法

✓需求测试
✓等价类测试
✓边界值测试
✓基于经验或知识的错误猜想测试

结构覆盖率度量方法

✓分支覆盖度测试
✓MC/DC覆盖度测试

➢ 软件集成验证和测试

验证方法

✓需求测试
✓接口测试
✓故障注入测试
✓数据流和控制流验证
…

测试用例选取方法

✓等价类测试
✓边界值测试
✓基于经验或知识的错误猜想测试
✓资源评估

结构覆盖率度量方法

✓功能覆盖度评估
✓函数调用覆盖度评估

…

➢ 嵌入式软件测试

测试环境

✓HIL测试
✓整车测试

测试方法

✓需求测试
✓故障注入测试

测试用例选取方法

✓等价类测试
✓边界值测试
✓基于经验或知识的错误猜想测试
…

Thank You

Momenta.ai

功能失效的风险来源

系统性故障

1. 确定性，可重复性
2. 可避免，可控制
3. 一般无法量化
…

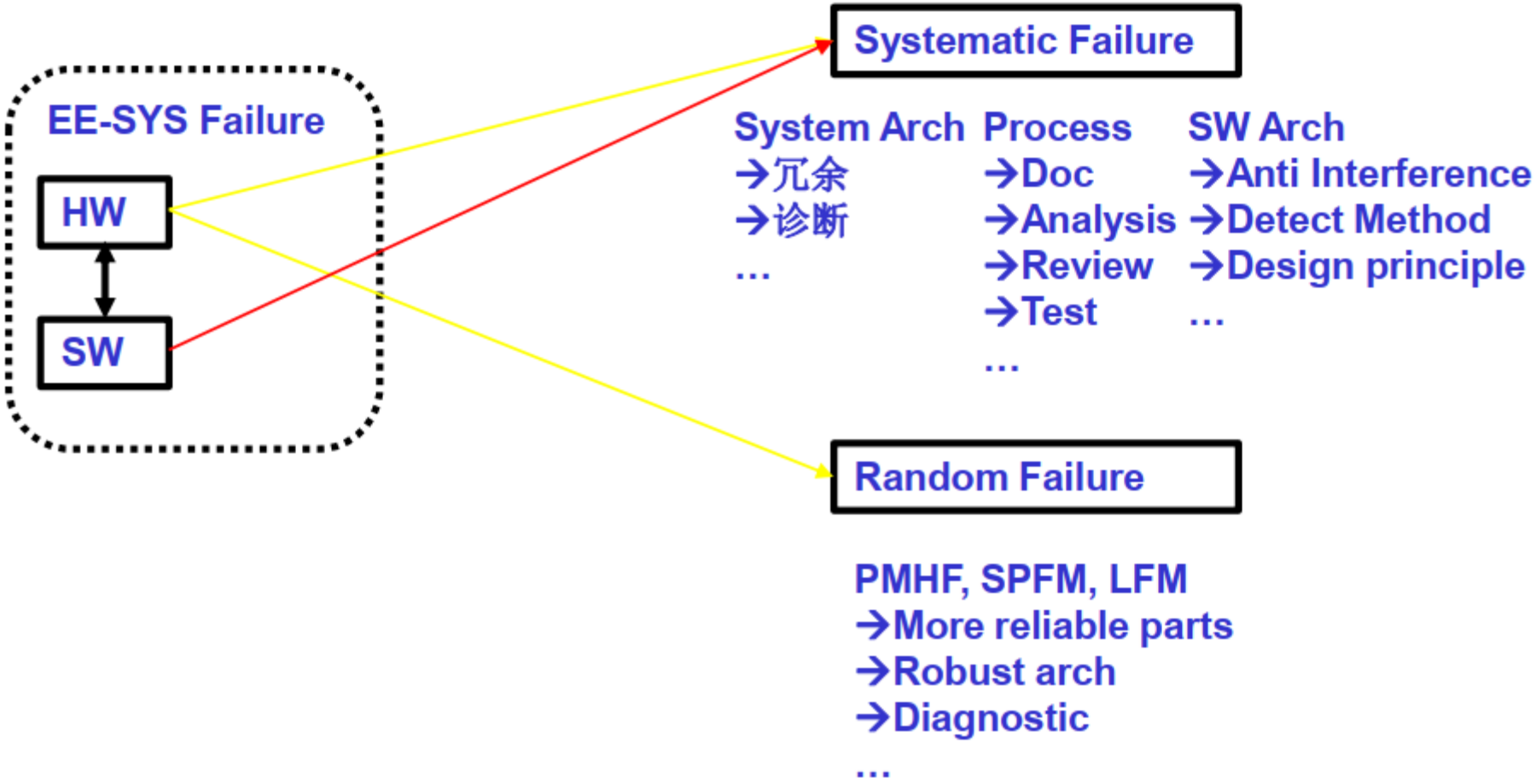
E.g.
1. 设计错误
2. Bugs
3. 制造错误
4. 操作错误
…

随机硬件失效

1. 不确定性，不可重复
2. 无法避免，可以控制
3. 可量化
…

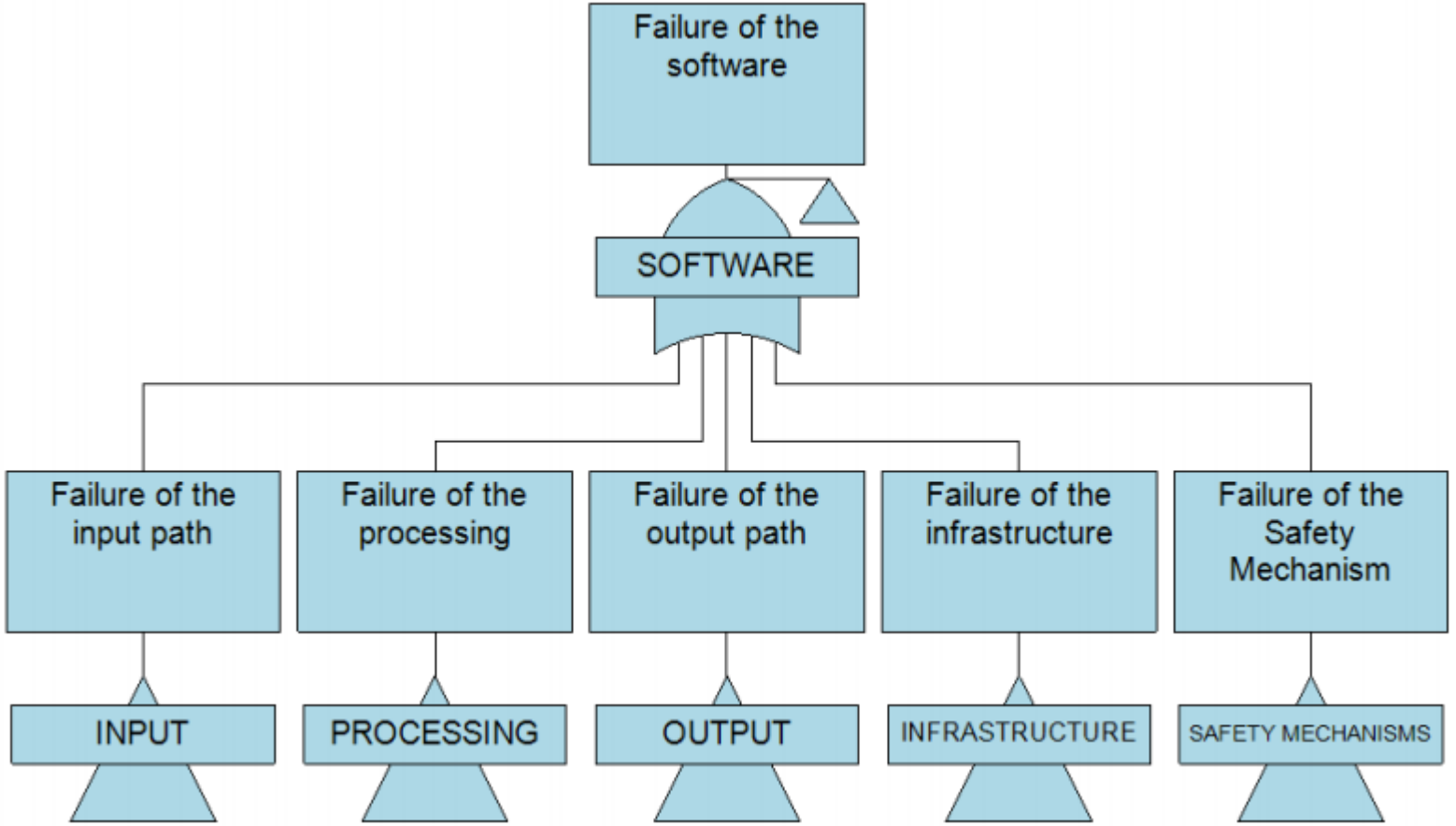
E.g.
1. MCU故障: 位翻转，软错误
2. 线路故障：开路/短路
3. 卡滞
…

功能失效应对措施



软件安全需求完整性

为保证软件需求的完整性，需充分考虑软件可能的失效模式，软件相关失效分类如下：



软硬件接口HSI

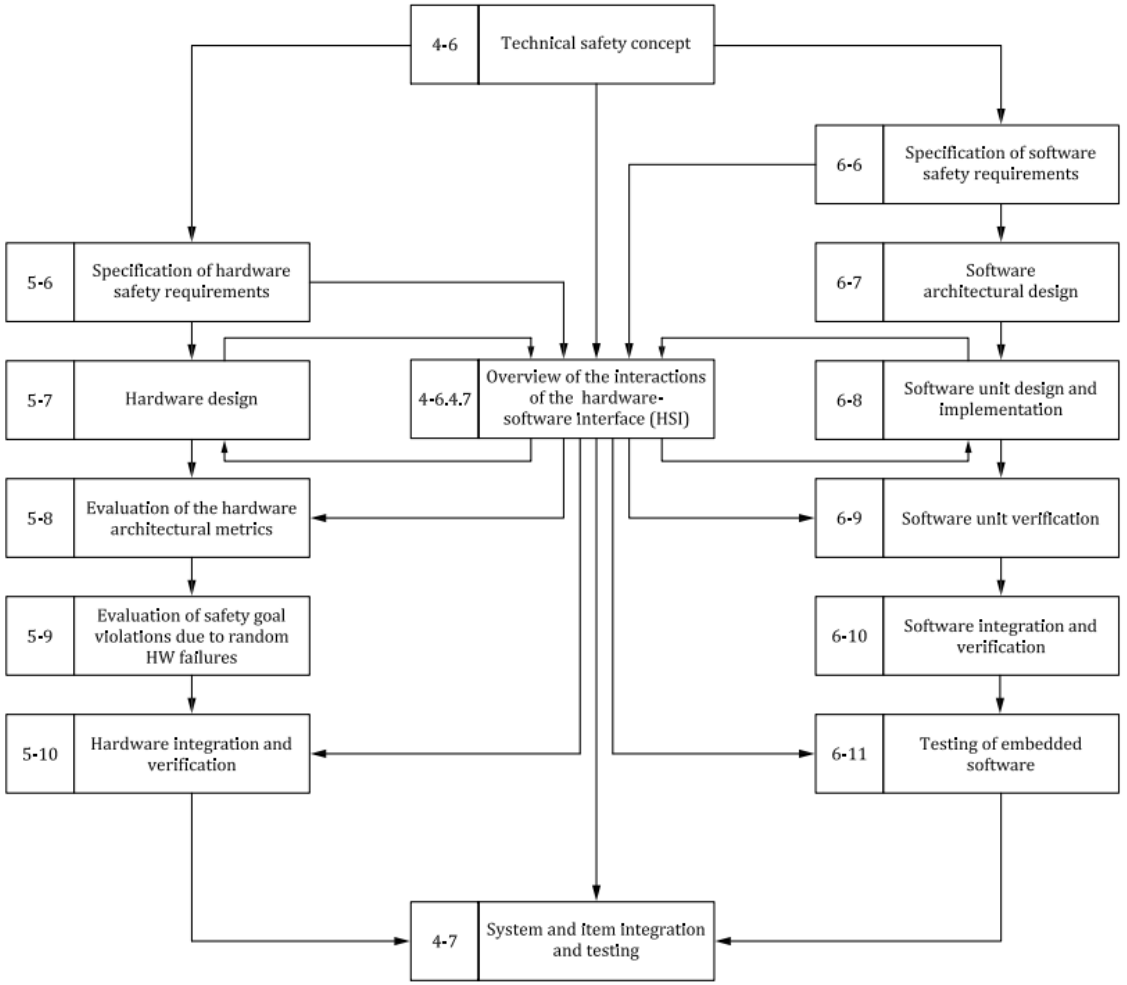


Figure B.1 — Overview of the interactions of the hardware-software interface (HSI)

软硬件接口HSI

软硬件接口元素

- Volatile Memory, e.g. RAM
- Non-Volatile Memory, e.g. NV-RAM
- BUS Interfaces, e.g. CAN, LIN
- A/D Converter
- D/A Converter
- PWM Modulation
- Multiplexer
- Electrical I/Os
- Watchdog
- Sensor Inputs
- …

软硬件接口特性

- Interrupts
- Timing consistency
- Data integrity
- Initialization (e.g. memory, registers, converter
- Message transfer
- Network modes (sleep mode, etc.)
- Memory management (address space, data ty
- Real time counter (e.g. start, stop, capture, com
- Sensor signal range (amplitude, dynamic)

相关失效诱因分类

Coupling factor class	Mapping to the topics in ISO 26262- 9:2018, 7.4.4	Examples at the software level
Shared Resource The same software, hardware, or system element instance is used by two elements, which are therefore affected by the failure or unavailability of that shared resource.	a) random hardware failures g) failures of common external resources or information	- SW component used by 2 other SW components, e.g. - maths or other libraries - I/O routines, drivers - Hardware resource used by more than one software element
Shared Information Input Connection to the same information source by means of which the two functions consume the same information, even in absence of shared resources, i.e. from a functional perspective.	a) random hardware failures	- Constants, or variables, being global to the two software functions - Data/function parameter arguments/messages delivered by software function to more than one other function
Insufficient Environmental Immunity Same or similar physical characteristics of elements, which can be affected by the same external environmental disturbance	f) environmental factors h) stress due to specific situation	Not directly applicable to SW alone. Environmental influences that affect behaviour of software can be considered at the system and hardware levels
Systematic Coupling Failure of elements due to a common systematic human error or tool error.	b) development faults c) manufacturing faults d) installation faults e) service faults h) stress due to specific situation	- Same software tools e.g. IDE, compiler, linker, software configurator - Same programming and/or modelling language - Same compiler/linker
Components of Identical Type Multiple instances of identical or very similar components can jointly fail due to a common cause failure.	a) random hardware failures b) development faults	- The same source code expanded twice, e.g. by usage of C macros NOTE: the same library instance or the same standard SW module instance called from different locations is rather considered as a Shared Resource.
Communication An element receives information from another element by means of a communication channel	a) random hardware failures b) development faults d) installation faults e) repair faults i) ageing and wear	- Data flow via global variables - Messaging - Function calls with arguments passed
Unintended Interface Two elements affecting each other directly via an unanticipated interface	a) random hardware failures b) development faults d) installation faults h) stress due to specific situation	- Same memory space which means a potential of wrong memory allocation or memory leaks