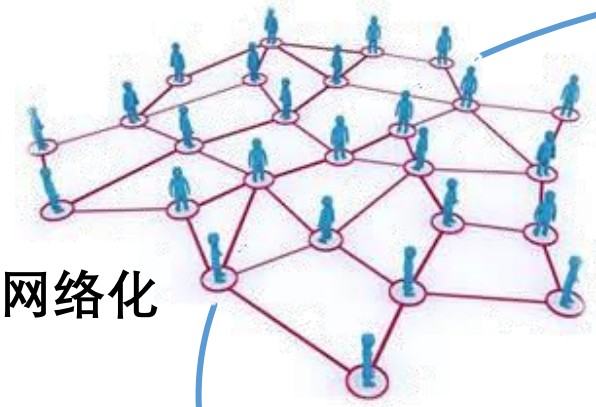
面向功能安全的嵌入式软件系统技术发展及挑战

吴际
北京航空航天大学计算机学院

报告提纲

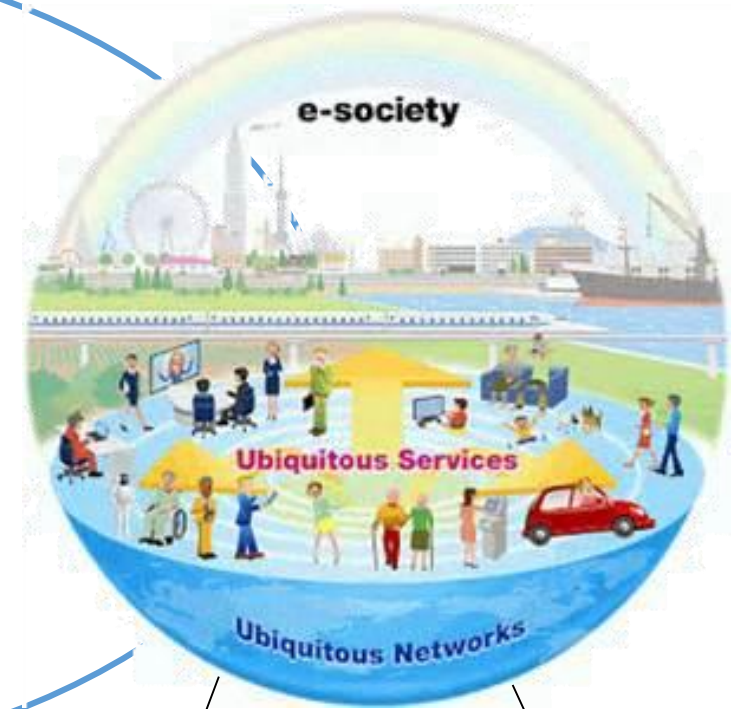
- 安全关键软件系统
- 如何满足安全要求
- 面向安全性的需求建模
- 面向安全性的体系架构设计
- 安全性挑战分析
- 总结

安全关键软件系统渗透到方方面面



网络化

智能化



泛在化

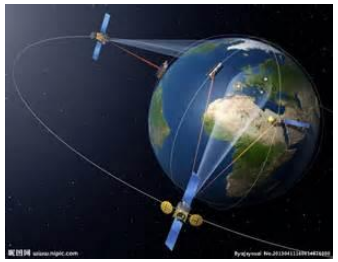
功能安全

信息安全

安全关键软件系统

- 广泛用于日常生活越来越依赖的飞机、高铁、汽车、能源、医疗等系统的控制和信息处理
- 国家十大科技发展规划中，至少5个都深度涉及安全关键软件系统
  - 航空发动机
  - 深海空间站
  - 智能机器人
  - 空间飞行器
  - 人脑工程



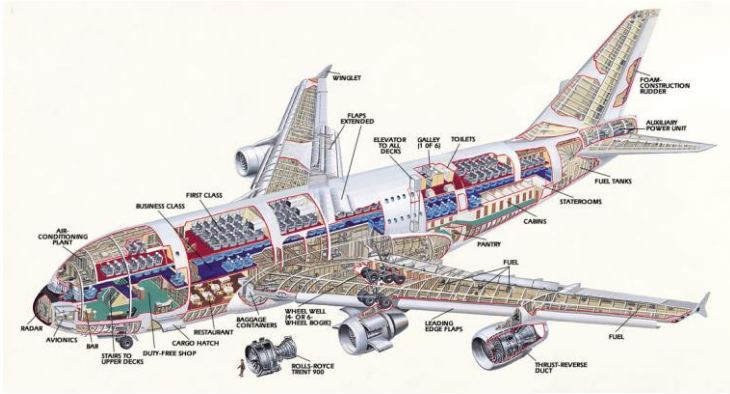


- 国外一直以来都非常重视，从国家层面部署了相应的研究计划
  - 欧盟FP6、FP7、Horizon2020
  - 美国NSF、NASA、DoD、DARPA、AFRL、FAA等
- 安全关键软件系统实现嵌入式系统的80%功能（甚至更多）

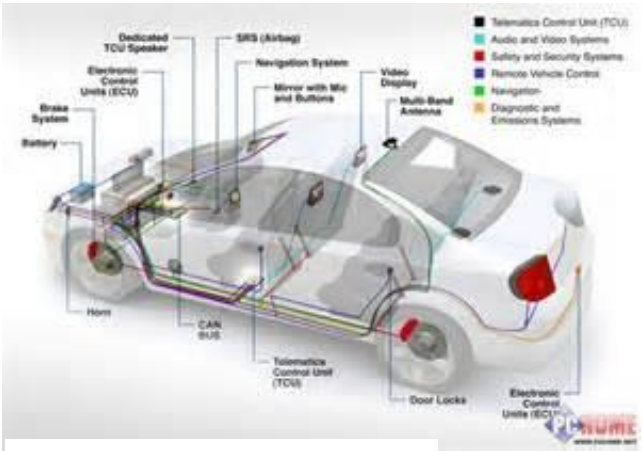
安全关键软件系统

- 失效会导致出现不可接受的安全事故，如人员伤亡、财产损失、环境危害等
- 在任何负载和运行状态下都要保证安全
- 跨工业领域部署
  - ✓ **机载飞行控制与管理系统/通讯系统/导航系统/...**
  - ✓ **航天载人与探测系统/飞行控制系统/...**
  - ✓ **人体健康监护系统/心脏监护/自动给药系统...**
  - ✓ **铁路交通控制系统/调度系统/...**
  - ✓ **汽车控制电子系统/汽车无人驾驶系统/驾驶稳定系统/...**
  - ✓ **核电安全控制系统/反应堆冷却系统/...**
- 典型特征
  - ✓ **异构节点的分布式系统，多种任务并发执行**
  - ✓ **对系统的响应时间和处理时序有严格要求**
  - ✓ **周期性任务和非周期性任务混合**

安全关键软件系统



- 传统机械控制系统等已逐渐被软件使能的电子系统取代
  - 空客A380、波音787上的软件系统超过500万行
  - 汽车系统的软件系统规模达到或超过1000万行
- 得益于电子和计算机技术的快速发展，系统性能指标得到了极大的提高
  - 飞机的燃油消耗、飞行舒适度、飞行安全性
  - 火车的行驶速度、稳定性和安全性
  - 汽车的安全性、驾驶稳定性等
- 近50%的计算机系统都部署和运行于安全关键系统中，需要对系统设计与实现进行详尽的审查和认证
  - 便捷和安全的交通服务
  - 医疗服务和健康监护
  - 能源转换、传输与控制
- 越来越强调系统化设计，平衡系统各个性能指标

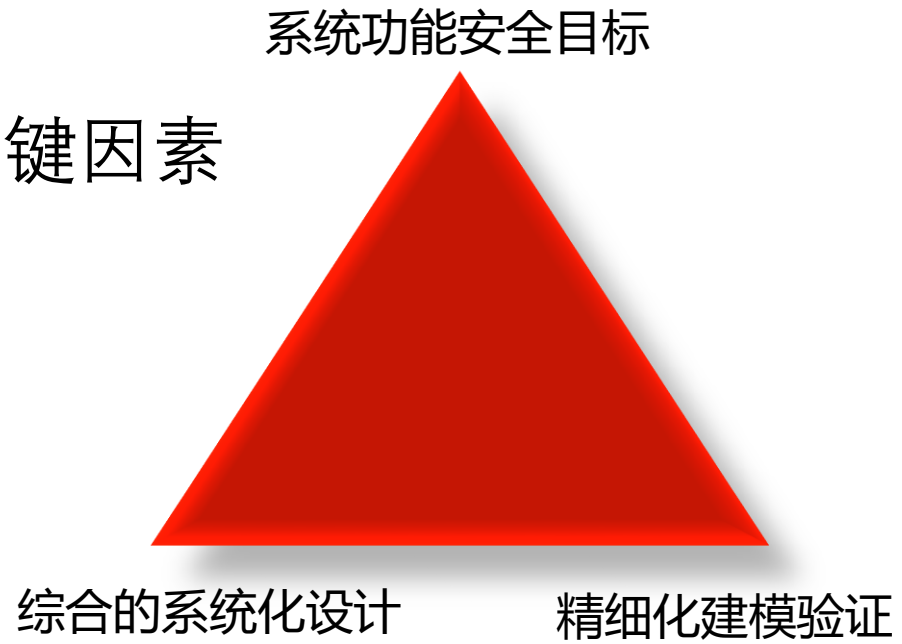


如何满足安全要求

- 为了确保嵌入式系统的功能安全(functional safety)，各个工业领域都制定了相应的强制性标准
  - 汽车领域：ISO26262
  - 航空领域：DO-178C、DO-297
  - 轨道领域：IEC61508
- 这些标准的共性特征
  - 强调从系统到软件，再到系统的综合性设计(ARP-4754)
  - 通过细致严格的过程来确保安全性(ARP-4761)
  - 强调精细化的需求分析和架构设计，并要求进行严格的验证
- 标准往往只在宏观层面提要求，不涉及具体技术措施

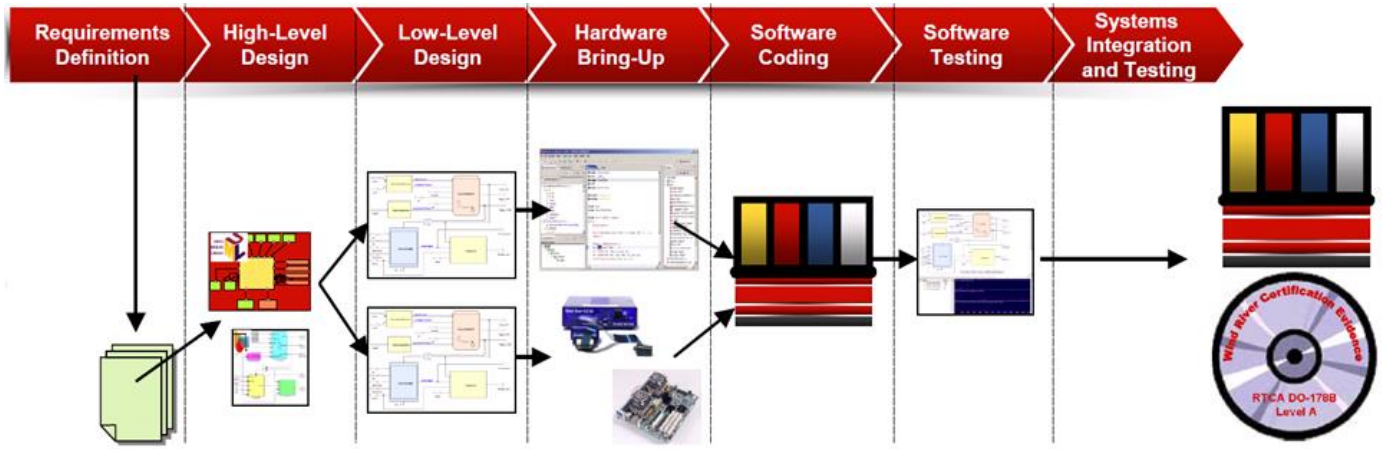
如何满足安全要求

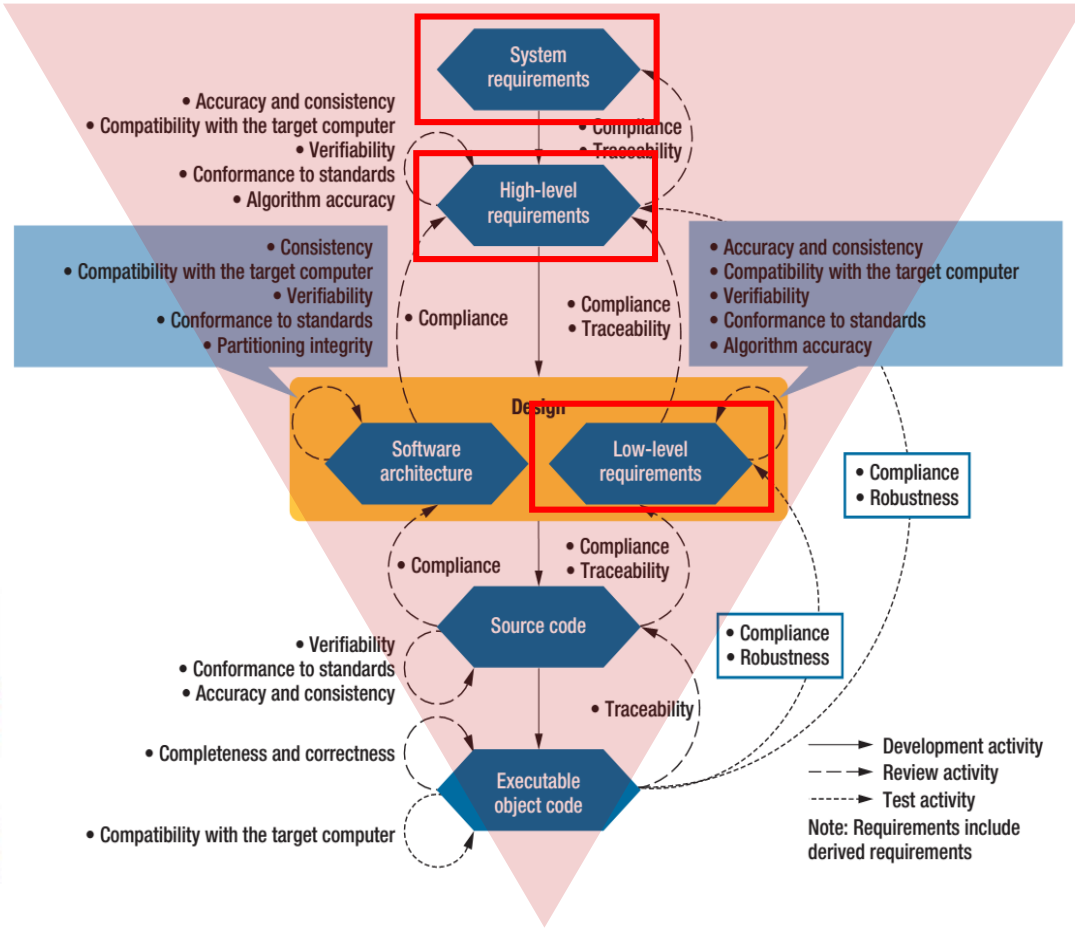
- 影响系统安全的因素众多，涉及多种技术的综合集成
  - 各种技术因素间相互影响
  - 系统和软件在需求和设计上需要协同
  - 要求：开展综合的系统化设计
- 需求与架构从来就是决定软件系统质量的关键因素
  - 从标准提取规范、规则和约束条件
  - 从领域提取设计共识，形成参考架构
  - 通过模型来描述需求和设计，形成层次化抽象
  - 要求：基于模型的构造与验证



航空安全关键软件的开发过程

- 三个层次的需求(DO-178B/C)
  - 系统需求, 软件高层需求, 软件低层需求
- 强调制品之间的追踪关系
- 强调验证



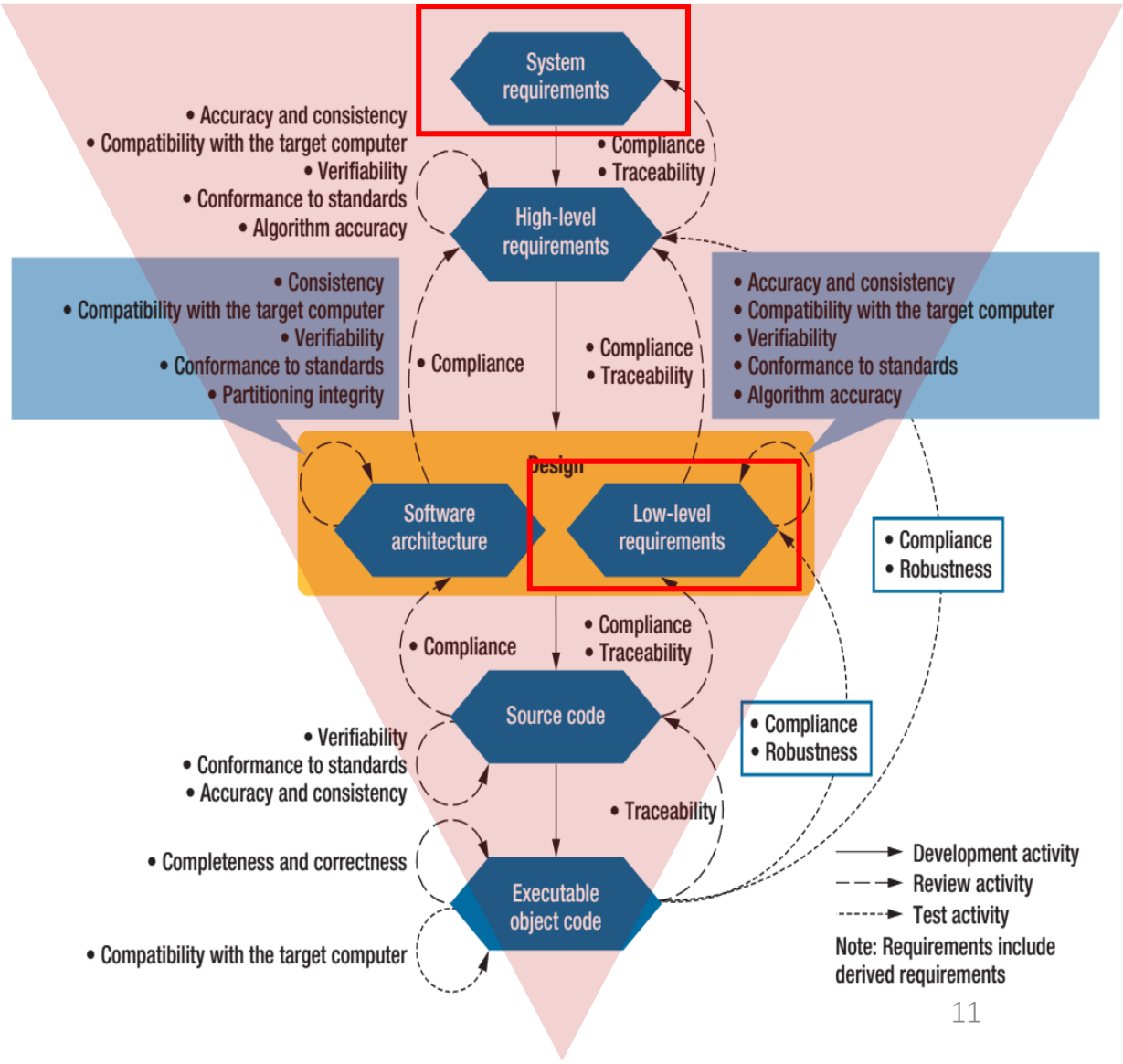


航空安全关键软件的需求

- 三个层次
  - 系统需求定义如何与外部实体进行交互，以及内部组件间的交互
  - 软件高层需求是系统分配给软件的需求
  - 软件低层需求是软件设计层次的需求
- 软件高层需求
  - 描述软件作为整体如何与外部环境进行交互，从而满足系统需求
- 软件低层需求
  - 描述一个软件单元如何与其外部使用者进行交互，从而满足高层需求和架构设计

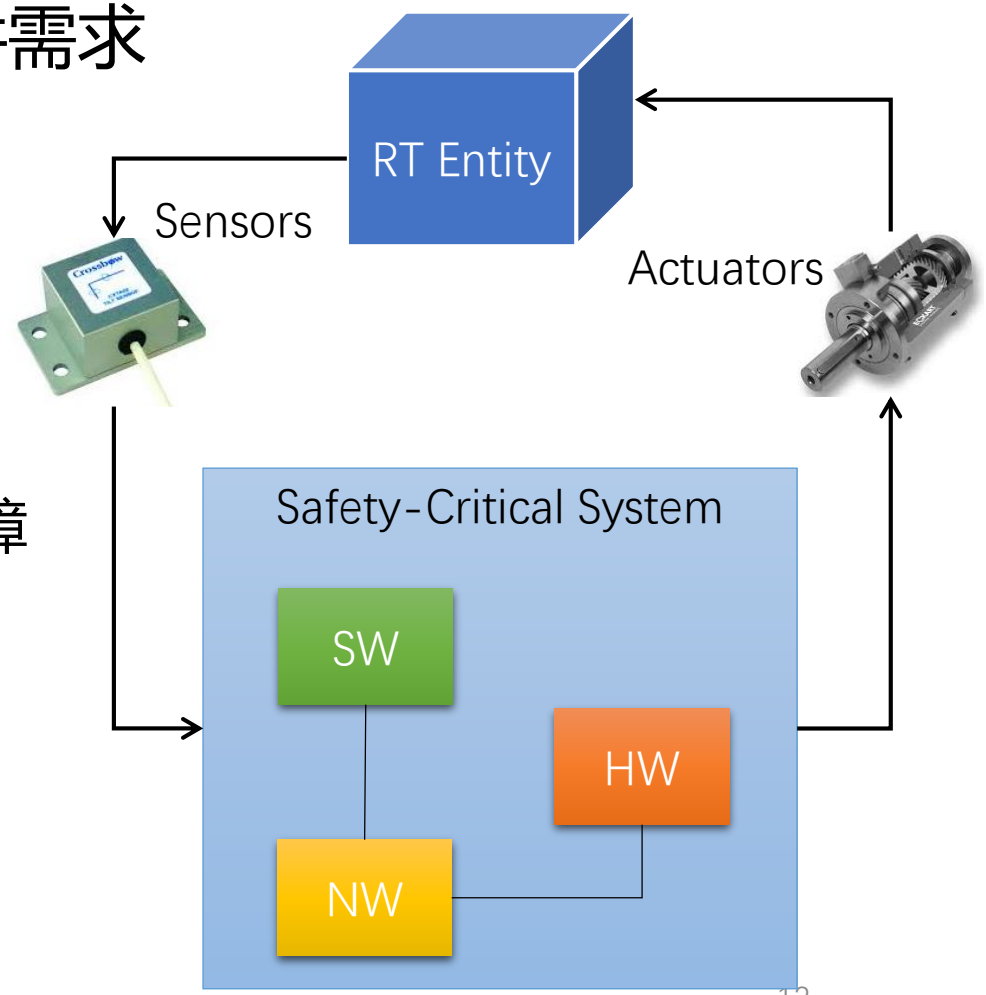
航空安全关键软件的需求

- 对需求质量有严格要求
  - 与上层需求关系的要求
    - 符合上层需求
    - 可追踪到上层需求
  - 特性要求
    - 准确、一致
    - 与目标计算机兼容
    - 可验证
    - 符合相关标准
    - 算法描述准确



安全关键软件的需求

- 必须要在系统上下文中来识别和分析其软件需求
- 系统与软件的关系
  - 系统为软件提供相应的资源
  - 系统定义了软件运行环境
- 软件需求识别
  - 关注RT实体的时间特性、状态特性
  - 关注系统内不同组件间的交互
  - 关注RT实体、系统和软件运行时可能出现的故障
- 需求表示
  - 基于文档(自然语言)
  - 基于规格模板(框架语言)
  - 基于模型(部分严格逻辑语言)
  - 基于形式化方法(严格逻辑语言)



安全关键软件的需求

- 文档化的需求表示
  - 易于管理和表达
  - 难以满足标准要求的准确、一致、无二义、可验证等要求
- 模型化的需求表示
  - 受制于具体模型语言和工具支持
  - 不同系统可能使用不同的模型语言，难以整合
- 规格化的需求表示
  - 需求描述边界严格定义(前置条件、后置条件和流程)
  - 往往采用模型化语言
- 形式化的需求表示
  - 采用严格的逻辑语言，表达严谨，不会有二义性
  - 技术难度大，不易掌握

规格化的需求描述-Case Study

- 需求规格模板定义了需要识别和描述的需求要素
  - Precondition (对用户的要求)
  - 从用户、环境或设备获得输入
    - 涉及输入机制设计
  - 确认所获得输入的有效性
    - 值和时间
  - 对输入的处理及其约束
    - 计算、转换、传输…
    - 故障检测与处理
  - 把处理结果发送给用户或环境
    - 涉及输出机制设计
  - Postcondition (对用户的承诺)

三类约束

- 时间约束
  - 截止期Deadline
  - 延迟Delay
  - 在确定时间段或时间点发生的事件
- 资源约束
  - 资源容量Volume
  - 资源可用性Usability
- 故障处理约束
  - 屏蔽、报告、处理

面向安全性的架构设计

- 基于组件的设计是目前普遍采用的方法
  - 组件：在功能和资源上相对独立的一个计算单元
  - 组件依赖关系
  - 组件接口
- 什么是架构？
  - 为解决系统的安全性问题而识别的一组核心组件
- 安全架构的约束
  - 必须限制故障传播
  - 必须采取主动措施去检测故障和容忍故障
  - 即便出现故障，也不会导致安全问题(fail-safe)

面向安全性的架构设计

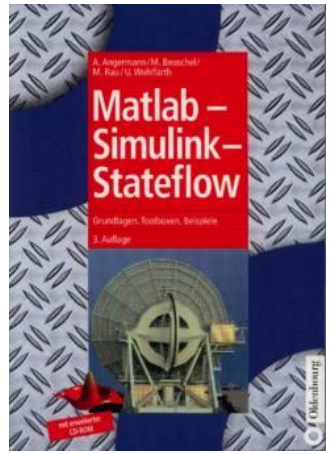
- 降低耦合确保组件的独立性
  - 减少互相影响
  - 消息通讯而不是同步调用或数据共享
  - 分区而不是公共资源池共享
- 故障限制区确保行为受控，即便发生故障
  - 组件设计为故障限制区(FCR)，限制内部故障向外部传播
  - 冗余结构(TMR)
- 行为确定性确保性能可预测
  - 组件间通讯的确定化：时间触发而不是事件触发
  - 资源分配的静态化：使用静态配置方式来确定资源分配
  - 减少动态性：谨慎使用诸如缓冲区、队列等设计结构

面向安全性的架构设计

- 关注点分离(separation-of-concern)的组件接口设计
  - 常规功能接口
  - 安全功能接口
  - 监控接口
  - 配置接口
  - 诊断接口
  - …
- 接口规格明确定义
  - 前置条件
  - 后置条件
- 接口权限控制
  - 不同类别的接口只限于特定类型的组件才能使用

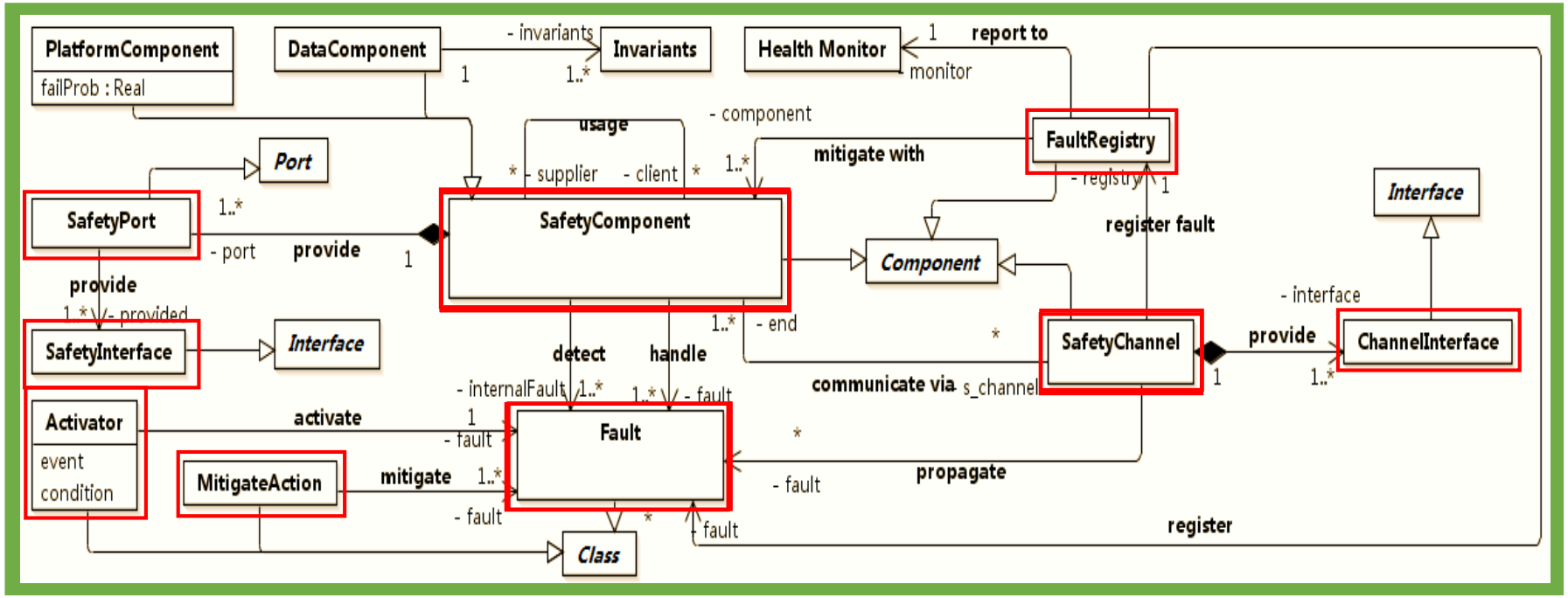
面向安全性的架构设计

- 使用标准化的建模语言及扩展机制进行设计
  - UML/sysML--MARTE
  - AADL
  - Simulink StateFlow
- 多视角模型的综合及验证
  - 组件、接口
  - 组件关系
  - 故障及检测机制
  - 故障处理机制
  - 相关约束



面向安全性的架构设计—Case study

安全体系结构模式



ID	关于故障处理的安全性要求
1	任何识别出来的故障，至少要求有一个安全组件对其进行检测和处理.
2	任何识别出来的故障都必须要描述其触发规则和处理策略.
3	对于可以传播的故障，一定要有其他的安全组件进行处理.
4	只有安全交互通道 (safety channel)能够把检测出的故障向故障注册模块报告.
5	只有故障注册模块可以根据预先设计好的故障处理策略来统筹所有组件进行故障处理.

面临的安全性挑战

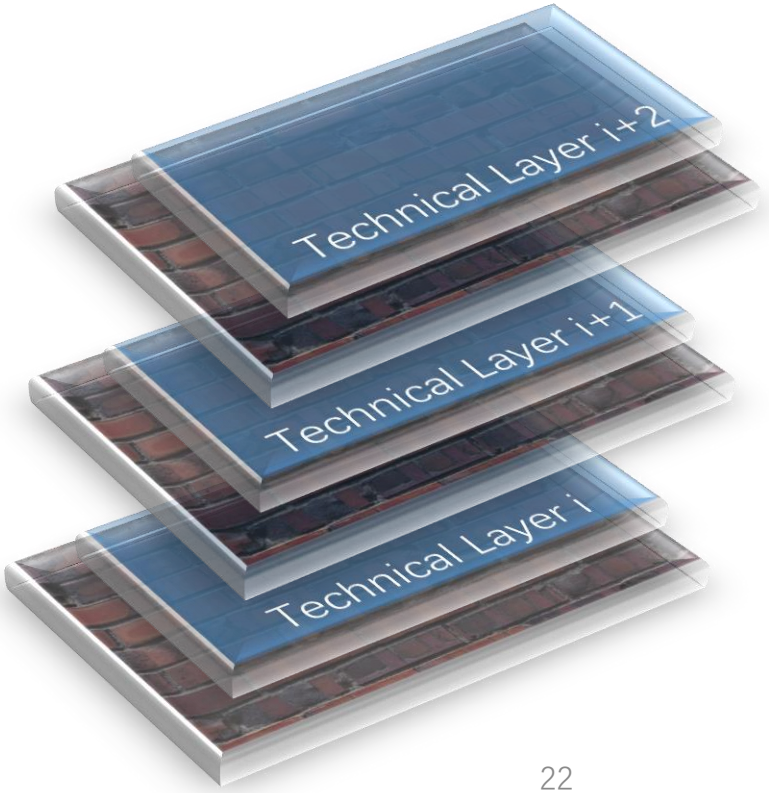
- 多个系统层次综合决定系统的功能和性能，如何跨越多个层次来综合设计以满足系统要求
- 不同性能之间具有竞争关系，如何系统化的开展平衡设计
- 功能安全与信息安全开始融合，必须要从故障(fault)与威胁(threat)两个角度综合开展设计
- AI开始在嵌入式系统中广泛应用，其不确定性对安全性有害

跨层次系统化设计挑战

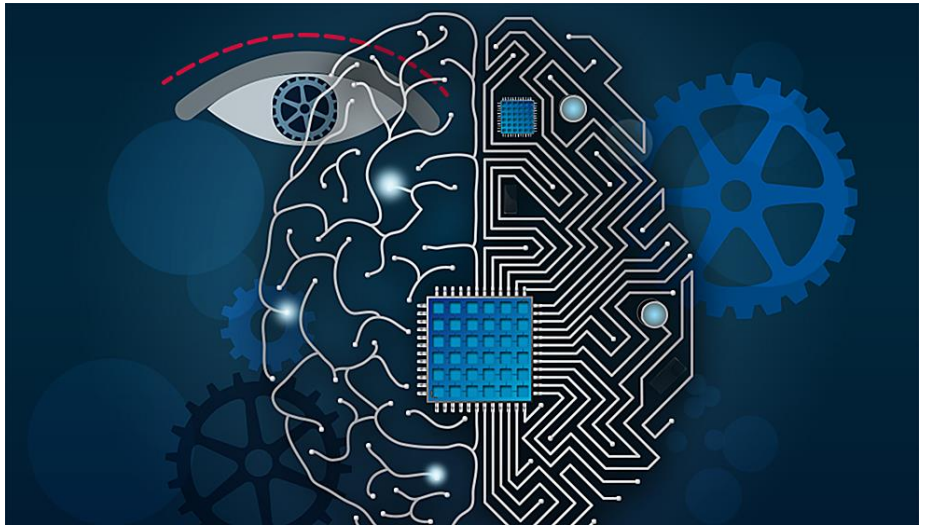
- 嵌入式系统具有层次性
  - **应用层：软件+数据**
  - **运行平台层：操作系统+通讯服务+任务调度服务+系统配置+编译环境+库函数**
  - **运行平台硬件层：系统硬件模块（含CPU、网络、内存等）**
  - **运行环境硬件层：外围I/O设备（含传感器、作动器等）**
- 系统具有涌现特性：源于不同层次间不同组件之间的大量交互
  - **失效概率（如<=10^{-9}）**
  - **系统失效模式**
  - **任务运行时间**
- 要求系统行为特性具有确定性
  - **时间确定性：在任何状态下，任务执行的时间特性都确定**
  - **故障处理确定性：在任何情况下，故障限制单元范围确定，故障处理效果确定**

跨层次系统化设计挑战

- 下层为上层提供资源池和服务
- 下层和上层之间具有独立的技术路线
- 传统嵌入式系统的三个假设
  - 下层技术假设不知道上层的使用模式和失效模式
  - 上层技术假设下层只为自己提供服务
  - 上层技术假设下层任何时候都以一致的方式来响自己的请求
- 如何来松弛这些假设，开展跨层次系统设计是主要挑战
  - 应用的分析与设计必须要结合OS提供的服务和约束
  - OS必须了解应用的具体需求来配置资源



AI带来的安全性挑战



- AI开始成为嵌入式系统的pervasive feature
- AI能够对传感数据进行更为灵活的处理
- 安全关键软件和系统最终也会使用AI技术
- AI对系统安全性的不利因素
  - AI模型(参数)本身会随着处理的数据不同而演化
  - AI模型对系统行为的影响具有不确定性
  - AI模型对系统/组件的失效模式影响未知
  - AI模型对系统的实时性影响具有不确定性
  - AI模型与引擎本身就可能存在漏洞可供利用

AI带来的安全性挑战

- 应对AI挑战的建议
  - **应用范围**：区分安全关键功能组件和非安全关键功能组件
    - 只在非安全关键功能组件中使用AI
    - 限制安全关键功能组件与非安全关键功能组件之间的交叉影响
  - **需求分析**：需要一套方法来描述AI引入的不确定性(uncertainty)
  - **设计方法**：需要为应用AI的组件专门定义一种接口和机制来检查/测试模型(参数)是否有缺陷，一旦发现问题可以reset模型(参数)
  - **测试方法**：需要一套方法限制AI对系统行为确定性带来的影响
  - **运行时机制**：如果要提供对AI模型(参数)进行动态(远程)更新机制，更新必须使用独立的隔离接口来完成

总结

- 安全关键系统应用广泛
  - 分布式
  - 实时性与可靠性要求高
  - 失效会带来严重安全后果
  - 开始深入应用AI
- 需求分析与体系结构设计是关键阶段
- 跨层次设计和AI是目前的主要挑战，也是技术发展驱动力