 DOI: https://doi.org/10.12349/iser.v6i8.7034

Research and Development of Electronic and Electrical Architecture Based on MBSE

Shuangyi Yang

Zhengzhou University of Industrial Technology, Zhengzhou, Henan, 475500, China

Abstract

With the transformation of the automotive, aerospace, high-end equipment and other fields towards "intelligence, electrification and networking", the electronic and electrical (EE) architecture has evolved from the traditional "distributed control" to "domain centralized central computing". Against this backdrop, this paper focuses on the integration path of MBSE and EE architecture development. Firstly, it sorts out the core theories of MBSE and EE architectures. Then, it constructs a four-stage MBSE development process of "requirements, functions, logic, and physics", verifies the application effectiveness with cases of intelligent vehicle EE architecture, and finally analyzes the current implementation challenges and proposes countermeasures. It aims to provide replicable MBSE practice solutions for the EE architecture development of complex equipment, promoting the transformation from "document-driven" to "model-driven".

Keywords

MBSE "Electronics and Electrical Engineering; Architecture development

基于 MBSE 的电子电气架构开发与应用研究

杨双义

郑州工业应用技术学院，中国 · 河南 郑州 475500

摘　要

随着汽车、航空航天、高端装备等领域向“智能化、电动化、网联化”转型，电子电气（EE）架构已从传统“分布式控制”向“域集中中央计算”演进。在此背景下，本文聚焦MBSE与EE架构开发的融合路径，首先梳理MBSE与EE架构的核心理论，随后构建“需求功能逻辑物理”四阶段MBSE开发流程，结合智能汽车EE架构案例验证应用成效，最后分析当前落地挑战并提出对策，旨在为复杂装备EE架构开发提供可复制的MBSE实践方案，推动“文档驱动”向“模型驱动”的转型。

关键词

MBSE；电子电气；架构开发

1 引言

基于模型的系统工程（MBSE）以“模型作为单一数据源”为核心，通过图形化建模、多域仿真与全流程追溯，实现系统从需求到设计、验证的一体化管理，为复杂 EE 架构开发提供了技术范式。国际汽车工程师学会（SAE）在《汽车 EE 架构开发指南》中明确提出，MBSE 可使 EE 架构开发周期缩短 25%30%，早期设计错误检出率提升 40% 以上。当前，宝马、特斯拉、空客等企业已将 MBSE 应用于 EE 架构开发，例如特斯拉 Model3 的中央计算 EE 架构通过 MBSE 方法实现需求功能硬件的全链路追溯，开发周期较 ModelS 缩短 40%。

【作者简介】杨双义（1987–），男，中国河南开封人，硕士，中级实验师，从事自动化控制研究。

2 MBSE 与 EE 架构开发的融合必要性

MBSE 与 EE 架构开发的融合，本质是通过“模型化”解决 EE 架构的“复杂度管理”与“全流程协同”问题，其必要性体现在三方面：

2.1 复杂功能的可视化建模

随着 EE 架构功能模块向“感知 - 决策 - 控制 - 执行”多域耦合发展（如智能汽车多传感器融合需联动激光雷达、域控制器、转向 ECU 等 10+ 硬件模块，涉及 CANFD、以太网等 5+ 通信协议），传统“需求文档 + 流程图 + 接口表”模式已失效。例如描述“激光雷达数据传输”时，自然语言“实时传输”未明确 10ms 或 50ms 时延阈值，易导致硬件团队按 50ms 设计通信链路，软件团队按 10ms 开发算法，引发后期集成冲突；且流程图无法关联波特率、信号 ID 等接口参数，工程师需在多份文档中交叉查询，效率低下。

MBSE 通过 SysML、UML 等标准化建模语言，构建“用例图 - 活动图 - 序列图”多视图模型：用例图明确“自动驾

驶决策”的参与者与核心场景（正常感知、故障冗余）；活动图拆解“目标识别”为“数据采集→噪声过滤→特征提取→目标分类”步骤，标注输入（激光雷达点云数据）、输出（目标位置）及高斯滤波等算法公式；SysML 序列图则以时间轴直观展示“激光雷达每 10ms 发点云数据→域控制器 20ms 内融合→决策 ECU5ms 生成指令→转向 ECU10ms 响应”的时序链，同步标注信号 ID=0x123 等参数与故障切换逻辑。这种建模方式让隐性逻辑显性化，且每个模型元素可追溯至原始需求（如数据采集时延≤ 10ms 对应 R-001 条款）。

2.2 多团队的协同开发

EE 架构开发需系统、硬件、软件、测试多团队协同，传统“文档驱动”模式存在三大壁垒：格式上，系统团队用 Word、硬件团队用 CAD、软件团队用 Visio，接口变更需手动同步，易出现“硬件改接口、软件用旧版”的兼容问题；理解上，“总线可靠性≥ 99.9%”需求，硬件关注阻抗匹配、软件关注协议重传、测试关注错误帧注入，缺乏统一标准；版本上，人工标注“V2.1”“V3.0”易导致多团队修改冲突。

MBSE 以“统一模型”为协同载体，通过 Siemens Polarion 等工具定义“需求 - 功能 - 逻辑 - 物理”四层元模型，明确 ECU 模块需关联功能模块与接口参数（如转向 ECU 关联转向控制功能，标注 12V 供电、CANFD 协议）。基于 Teamcenter 云端平台，各团队可实时协同：系统团队完成功能建模后，硬件团队直接查看域控制器“以太网带宽≥ 1Gbps”需求；软件团队基于活动图开发算法，若系统新增“雨天感知补偿”，模型平台自动推送变更通知；测试团队根据用例图生成“断开激光雷达验证冗余”的测试用例，且用例与模型实时联动。

2.3 早期设计的仿真验证

传统 EE 架构开发遵循“设计→试制→验证”线性流程，关键验证需等样件下线，风险后置代价极高。如智能汽车总线设计中，硬件团队凭经验估算负载 70%，但实际多 ECU 并发传输时，峰值负载可能超 80% 安全阈值，此时修改需重新制作 PCB 板，成本超 100 万元，延误 2-3 个月。

MBSE 通过模型与仿真工具集成，实现设计阶段虚拟验证：物理架构设计时，将含 ECU 参数、总线拓扑的 MBSE 模型导入 VectorCANoe，仿真不同工况下总线负载——若自动驾驶模式中激光雷达 10ms 发 1000 字节数据导致负载达 85%，可在模型中拆分数据（200 字节关键数据 10ms 发，800 字节非关键数据 50ms 发），重新仿真负载降至 72%；功能逻辑验证时，将活动图转化为 Matlab/Simulink 模型，如船舶应急供电功能仿真中，可提前发现应急发电机启动超时问题并优化控制算法。

3 基于 MBSE 的 EE 架构开发流程与应用

基于 MBSE 的 EE 架构开发遵循“需求驱动、模型迭代、仿真验证”的原则，分为需求分析与建模、功能建模、逻辑架构建模、物理架构建模、验证与优化五个阶段，各阶段通过模型追溯形成闭环，具体流程如下：

3.1 阶段 1：需求分析与建模——明确“做什么”

需求是 EE 架构开发的起点，本阶段通过 MBSE 实现需求的结构化、可追溯化管理，核心任务包括需求获取、需求建模与需求验证：

1. 需求获取：通过用户访谈、行业标准（如 ISO26262 功能安全标准）、竞品分析，梳理 EE 架构的三类需求：功能需求：如“支持 L2+ 级自动驾驶，包含 AEB、ACC 功能”；非功能需求：如“总线响应延迟≤ 100ms，硬件 MTBF（平均无故障时间）≥ 10000h”；约束需求：如“硬件成本≤ 5000 元 / 车，兼容未来 OTA 升级”。

2. 需求建模：采用 SysML 需求图（Requirement Diagram）与用例图（UseCase Diagram）构建需求模型，以智能汽车 EE 架构为例：用需求图分层管理需求：顶层为“整车 EE 架构需求”，中层为“域控制器需求”（如自动驾驶域、智能座舱域），底层为“ECU 需求”（如感知 ECU 的算力≥ 200TOPS）；用追溯关系（Trace）关联需求与用例：例如“AEB 功能需求”追溯至“用户触发 AEB”用例，确保需求与用户场景一致；输出《需求模型报告》，包含需求清单、追溯矩阵（需求用例功能的映射关系）。

3. 需求验证：通过“需求评审 + 仿真预验证”确保需求可行性：组织系统、硬件、测试团队开展需求评审，检查需求的完整性（无遗漏）、一致性（无冲突）、可验证性（如“总线响应延迟≤ 100ms”可通过仿真验证）；对关键非功能需求（如总线延迟），用 Matlab/Simulink 搭建简化仿真模型，预验证需求是否可实现，避免后期设计偏离。

3.2 阶段 2：功能建模——拆解“怎么做”

本阶段将需求转化为可执行的功能模块，通过 MBSE 实现功能的可视化分解与交互定义，核心任务包括功能分解、功能流程建模与功能分配：

1. 功能分解：基于需求模型，采用“自顶向下”的分解方法，将顶层功能拆解为子功能，以智能汽车“自动驾驶功能”为例：顶层功能：自动驾驶功能；一级子功能：环境感知、决策规划、控制执行；二级子功能：环境感知包含“激光雷达感知、毫米波雷达感知、摄像头感知”，决策规划包含“路径规划、行为决策”。

2. 功能流程建模：用 SysML 活动图（Activity Diagram）描述功能间的数据流与控制流，例如“AEB 功能流程”：起始节点：传感器（激光雷达 + 毫米波雷达）采集障碍物数据；活动步骤：感知模块融合数据→决策模块判断碰撞风险→控制模块触发制动；分支条件：若碰撞时间（TTC）≤ 1.5s，触发紧急制动；若 TTC ＞ 1.5s，保持正常行驶；

3. 功能分配：将子功能分配至对应的域（如自动驾驶域、车身域），用 SysML 内部块图（Internal Block Diagram）

标记功能归属——例如“激光雷达感知”分配至自动驾驶域，“车窗控制”分配至车身域，为后续逻辑架构建模奠定基础。

3.3 阶段 3：逻辑架构建模——设计“模块与接口”

1. 逻辑模块划分：基于功能分配结果，遵循“高内聚、低耦合”原则划分逻辑模块，以智能汽车 EE 架构为例：自动驾驶域逻辑模块：感知模块、决策模块、控制模块、传感器管理模块；智能座舱域逻辑模块：显示控制模块、语音交互模块、导航模块；用 SysML 块定义图描述模块属性（如感知模块的“数据处理时延≤ 50ms”），确保模块功能与子功能一一对应。

2. 接口定义：明确逻辑模块间的接口类型（数据接口、控制接口）与协议，例如：感知模块→决策模块的接口：数据类型为“障碍物位置、速度”，协议为“以太网 SOME/IP”，传输频率为 10Hz；决策模块→控制模块的接口：控制信号为“制动请求、转向角度”，协议为“CANFD”，传输延迟≤ 20ms；用 SysML 接口图（Interface Diagram）记录接口参数，形成《逻辑模块接口规范》，避免后期硬件与软件接口不匹配。

3. 交互建模：用 SysML 序列图（Sequence Diagram）描述逻辑模块的交互时序，例如“ACC 功能交互”：时序步骤：传感器管理模块→感知模块（发送障碍物数据）→决策模块（发送跟车距离指令）→控制模块（发送油门 / 制动信号）；标注关键时序指标（如“感知模块→决策模块的传输时延≤ 30ms”），为后续物理架构的总线设计提供依据。

3.4 阶段 4：物理架构建模——落地“硬件与拓扑”

本阶段将逻辑模块映射至具体硬件，设计硬件拓扑与总线网络，核心任务包括硬件选型、拓扑设计与总线仿真：

1. 硬件选型与映射：根据逻辑模块的性能需求（如算力、接口数量），选择对应的硬件（ECU、传感器、总线），并建立“逻辑模块硬件”的映射关系：感知模块→自动驾驶域控制器（算力≥ 500TOPS，支持以太网接口）；显示控制模块→智能座舱域控制器（支持 HDMI2.1 接口，显存≥ 4GB）；用 SysML 物理图（Physical Diagram）展示映射关系，确保硬件性能满足逻辑模块需求（如域控制器算力≥逻辑模块的算力需求 120%，预留扩展空间）。

2. 硬件拓扑设计：设计 ECU、传感器的物理布局与总线拓扑，以智能汽车为例：总线选型：自动驾驶域采用车载以太网（传输速率 1000Mbps），车身域采用 CANFD（传输速率 8Mbps）；拓扑结构：采用“中央网关 + 域控制器”架构，中央网关连接自动驾驶域、智能座舱域、车身域，实现跨域数据交互；用 Auto CAD Electrical 绘制硬件拓扑图，标注硬件安装位置（如自动驾驶域控制器部署于后备箱）、总线走向（如以太网布线避开高压线束）。

3. 总线仿真验证：用专业工具（如 CANoe、Vector CANalyzer）仿真总线性能，验证拓扑设计可行性：

总线负载仿真：模拟峰值场景（如自动驾驶域同时传输激光雷达、摄像头数据），计算总线负载，确保峰值负载≤ 70%（预留 30% 冗余）；延迟仿真：测试跨域数据传输延迟（如智能座舱域→自动驾驶域的导航数据），确保延迟≤ 100ms；若仿真发现总线负载超标的问题（如某 CANFD 总线负载达 85%），则优化硬件映射（如将部分功能迁移至其他 ECU）或升级总线类型（如将 CANFD 改为以太网）。

3.5 阶段 5：验证与优化——闭环“迭代改进”

本阶段通过“模型仿真 + 物理测试”验证 EE 架构方案的满足性，形成“验证反馈优化”的闭环，核心任务包括模型验证、物理测试与方案优化：

1. 模型验证：基于 MBSE 工具链开展多维度仿真验证：需求验证：通过模型追溯工具（如 PTC Integrity）检查 EE 架构模型是否覆盖所有需求，需求满足率≥ 98%；功能验证：用 Simulink 仿真逻辑模块功能（如感知模块的障碍物识别准确率），确保功能达标（如准确率≥ 95%）；可靠性验证：用 ANSYS Simplorer 仿真硬件热设计，确保 ECU 工作温度≤ 85℃，避免过热故障。

2. 物理测试：制作 EE 架构原型（如域控制器原型、总线网络原型），开展物理测试：硬件性能测试：测试域控制器的算力、接口稳定性（如连续 72 小时满负荷运行无故障）；总线通信测试：用 CANoe 模拟实车场景，测试总线的抗干扰能力（如在电磁干扰环境下，总线误码率≤ 10^{-6}）；功能集成测试：将硬件原型与软件算法集成，测试端到端功能（如 AEB 功能的触发准确率）。

4 结论

MBSE 通过“模型作为单一数据源”，为复杂 EE 架构开发提供了“需求可追溯、设计可视化、验证早期化”的解决方案，其核心价值在于：1）打破信息孤岛，实现需求功能逻辑物理的全链路追溯，减少设计冲突；2）通过多域仿真提前验证方案可行性，降低后期返工成本；3）提升多团队协同效率，缩短开发周期。对于企业而言，落地 MBSE 需从“标准、工具、人才”三方面发力：先建立统一的建模规范，再通过工具链集成打破数据壁垒，最后培养复合型人才，逐步实现从“试点项目”到“全流程推广”的转型，让 MBSE 成为 EE 架构开发的核心竞争力。

参考文献

[1] 陈江波,罗峰,刘钦,等. 增程式电动汽车电子电气架构研究[J].汽车测试报告,2025,(09):22-24.

[2] 刘自凯,徐伟,姚昂,等. 汽车电子电气架构的功能架构设计研究[J].汽车电器,2025,(02):90-92.

[3] 董文亮. 智能网联汽车电子电气架构升级路径研究[J].汽车测试报告,2025,(02):25-27.