

基于 SOA 的中央式架构实践与挑战

程福明，刘聪，郝鹏
上海禾骋科技有限公司

【摘要】 博世关于电子电气架构三大阶段、六小阶段的演进路线在业内被广泛认可，软件定义汽车已成为行业共识和未来发展趋势。智能汽车的电子电气架构正由分布式过渡到（跨）域集中式往中央式集中式架构迈进，为实现软件定义汽车，智能汽车的软件架构正向面向服务的架构革新。本文分析了面向服务的中央式架构的特点、关键技术，描述了面向服务的中央集中式架构开发方法，总结了面向服务的中央集中式架构开发在实践过程中存在的问题和挑战，并给出了解决问题的方法。

【关键词】 面向服务的架构，中央集中式架构，软件定义汽车

Practice and Challenges of SOA and Centralized Architecture

Cheng Fuming, Liu Cong, Hao Peng
Shanghai He Cheng Technology Co., Ltd.

Abstract: Bosch's evolution path of the three major stages and six minor stages of electronic and electrical architecture has been recognized in the industry, and software defined vehicle have become an industry consensus and future development trend. The electrical and electronic architecture of smart cars is transitioning from distributed to cross domain centralized to centralized architecture. In order to realize the software definition of vehicle, the software architecture of intelligent vehicles is moving towards service-oriented architecture innovation. The article analyzes the characteristics and key technologies of service-oriented centralized architecture, describes the development methods of service-oriented centralized architecture, summarizes the problems and challenges in the practical process of service-oriented centralized architecture development, and provides solutions to these problems.

Key words: service-oriented architecture, centralized architecture, software defined vehicle

引言

用户追求汽车更多智驾、娱乐、网联功能，追求汽车个性化、私人订制、专属关怀以及软件功能持续进化带来的新鲜感，使得汽车的产品属性正从一个传统移动出行工具逐渐向智能移动终端加速转变^[1]，也为中国汽车产业向着电动化、网联化、智能化方向转型发展提出了更多的挑战与机遇。

传统汽车电子电气架构硬件架构由多个独立控制组成，软件架构开发面向信号的特点。传统的硬件架构方案存在硬件资源分散、面向信号的通信效率低、软硬件高度耦合、软件升级不便的问题；传统的嵌入式软件开发方式存在功能代码耦合、部署灵活性差、拓展性差、嵌入式的交叉编译机制导致软件更新难、功能重新分配导致关联的控制器软件需要同步更新引起整体开发难度大、周期长的问题。这种现状难以支撑当前的汽车软件开发规模、复杂度和迭代速度。这就要求需要构建一个可持续进化的电子电器架构和支持软件快速迭代升级的软件架构。诚然，基于面向服务的架构（Service-Oriented Architecture, SOA）的中央式架构方案被认为是解决这一问题的核心技术之一^[2]，在国内外各大OEM中最新研发架构平台上得到了逐步应用。在这项技术中，中央计算平台+区域控制器的电子电气架构提供了硬件基础，SOA 基于面向对象的特点，其面向服务开发的软件架构开发提供了软件基础。总结这套架构方案有如下几个几

项关键技术支撑：首先通过汽车开放系统架构（Automotive Open System Architecture, AUTOSAR）软件架构实现软硬分离，使得应用软件工程师可以隔离复杂的硬件专注于上层应用软件的开发；通过 SOA 中间件进一步标准化不同应用层软件的接口、使得应用层软件部署更灵活、集成更容易、调度更便捷，从而实现应用层软硬解耦，降低软硬耦合度；然后通过融合的中央集中式架构满足高算力的需求，解决带宽的瓶颈，提升安全性；其次基于面向对象开发、面向服务开发的软件架构开发方法，解决集成难度大，软件迭代成本高的问题；最后通过空中下载技术（Over-the-Air Technology, OTA）技术，满足用户的新鲜感。

SOA 作为一种面向对象开发、基于服务设计软件实现技术，其更是一种设计指导方法。本文分别 SOA 中央式架构的典型网络拓扑形态、SOA 中间件协议、软件的分层模型、服务的部署原则展开描述，最后综述一种正向的 SOA 中央式架构开发流程并结合开发实践描述存在的问题和给出的解决问题的方法。

1 中央式架构的特点

按照博世定义的电子电气架构演进路线，汽车电子电气架构的发展演进分为三个大阶段，它们分别是早在十年前就已经开始成熟的分布式电子电气架构、近五年发展起来的（跨）域集中电子电气架构和当前以及未来重点研究应用的中央集中式电子电气架构。中央式电子电气架构的实现又分为两个细分阶段，即当前全球各大主流汽车势力正在开

展研究与应用的中央计算平台+区域控制器阶段和未来的以云平台为中央计算平台补充的车-云一体计算的最高阶段。本文研究重点是第一个细分阶段的基于 SOA 的中央计算平台+区域控制器的实践，一种典型的中央式架构物理拓扑形态如图 1 所示。下面围绕该架构的特点展开介绍。

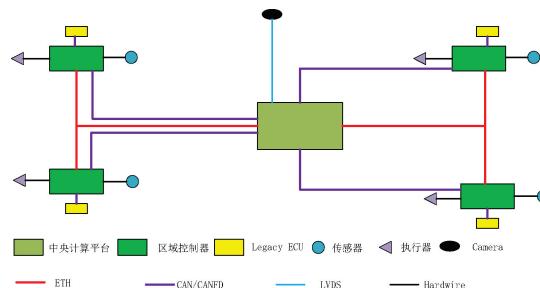


图 1 一种典型的中央式架构拓扑图

1.1 功能架构集中化

与传统分布式架构将功能的传感、处理、执行都分散在各个控制器上不同，传统分布式架构的控制器既是该功能的逻辑处理器，也集成了该功能的传感信号采集通道和该功能的执行器驱动能力；中央式架构功能分配策略上基于功能传感处理执行（Sensor Process Actor, SPA）模型将功能进行了两层分配，即功能 P（逻辑处理）层和功能 S/A（传感执行）层。中央计算平台属于 P 层（逻辑处理）集中部署整车的功能处理逻辑；区域控制器、基于其物理位置被直接采集的传感器、被直接驱动的执行器及被配电控制和数据路由的传统电子控制器单元（Electronic Control Unit, ECU）属于 S/A 层（传感执行）由中央计算平台进行统一传感信号输入调度与执行器输出控制。中央式架构功能分配模型如图 2 所示。

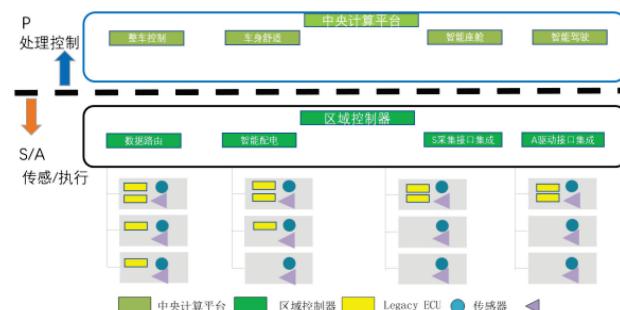


图 2 中央式架构功能分配模型图

中央式架构将车辆的功能逻辑集中上移至高性能、多资源的中央计算平台，其融合了整车控制功能、车身舒适功能、智能座舱功能和智能驾驶功能，实现了融合功能的统一调度及控制；中央式架构将车辆的执行器、传感器、传统的 ECU 根据就近原则连接到基于物理位置分布在车辆周围的区域控制器上，区域控制器实现基于物理位置布置的传感器、执行器的输入/输出接口集成、传统 ECU 的智能配电控制和作为传统 ECU 的网关实现通信数据的路由。

1.2 物理架构区域化、线束布局简化

根据中央式架构在功能分配上的分层策略，将功能逻辑

集中上移至中央计算平台，将功能的传感输入接口和执行输出接口就近接入按照物理位置布置的区域控制器的集成输入输出接口上，这种基于物理分布位置的区域控制器与传感执行器就近连接方式减少了线束的长度，这种功能逻辑集中上移至中央计算平台减少了整车的 ECU 数量。分别以分布式架构和中央式架构在车辆尾部增加一个高位制动灯功能为例，独立的车身控制模块（Body Control Module, BCM）和中央计算平台均布置在驾驶舱。分布式架构下，高位制动灯功能部署在 BCM 上，BCM 驱动高位制动灯的信号需要绕过车身跨过后部防火墙连接到高位制动灯；中央式架构由于基于物理位置布置的后部域控制器集成了执行器输出接口，只需配置一个输入输出（Input Output, I/O）口就近连接高位制动灯即可实现对高位制动的信号采集和点亮驱动，高位制动灯功能逻辑部署在中央计算平台。对比两种架构方案，中央式架构在该功能实现上节约了线束 80% 以上，二者对比图如图 3 所示。

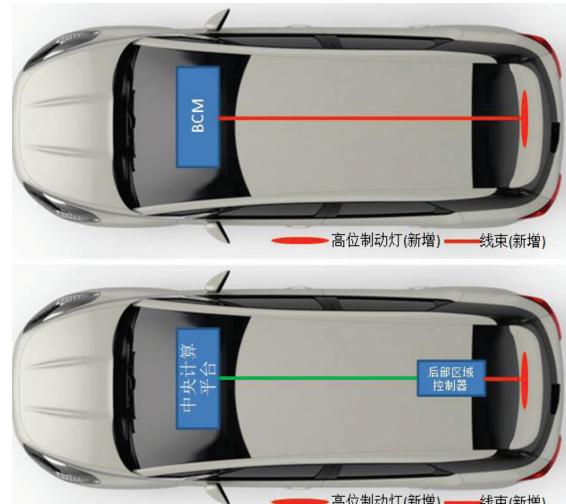


图 3 新增功能线束对比图

1.3 软件架构服务化、服务分层化

1.3.1 软件架构服务化

源自于 IT 行业的 SOA 技术自 20 世纪 90 年代被提出后，作为一种软件设计模式，SOA 被 IT 技术领域广泛应用并得到快速发展^[3]。组件开发和集成（Component Based Development and Integration, CBDI）论坛将 SOA 定义为“使应用功能能够以与服务消费者相关的粒度发布的服务集来提供和服务的政策、实践和框架。并使用单一的、基于标准的接口形式从实现中抽象出来^[4]。” SOA 面向对象开发、基于服务设计的这种理念被引入到汽车行业，IT 行业和汽车行业之间的技术差异在于，在汽车领域使用专门的以太网作为特定应用的物理层^[5]，使用基于 IP 的可扩展面向服务的中间件（Scalable service-Oriented Middleware over IP, SOME/IP）或数据分发服务（Data Distribution Service, DDS）作为专门的协议^{[6][7]}。基于 SOA 的理念，在整车软件架构设计中，汽车的产品能力被封装并抽象为服务，服务之间通过标准服务接口进行调用，服务访问通过服务总线建立连接，应用层可通过调用服务进行组合来实现，而不必重点关注内

部实施方案及硬件部署方案。这种基于服务的架构设计可实现软硬件解耦、功能的灵活组合与快速实施，这些特性解决了汽车架构设计面临的挑战，使得 SOA 架构得到了逐步的应用。

1.3.2 软件分层化

中国汽车工业协会软件定义汽车工作组在应用层软件上将服务分为设备抽象服务、原子服务、应用服务 3 层，这种服务软件分层是提升软件复用性、降低软件开发复杂度的关键。我们把应用服务、原子服务对应功能架构的 P（逻辑处理）层，设备抽象对应功能架构的 S/A（传感执行）层。软件定义汽车工作组基于车辆特性定义的服务分层、服务颗粒度划分建议图如图 4 所示。



图 4 服务分层、服务颗粒度划分建议图

设备抽象服务、原子服务、应用服务的特点分别是：

1) 设备抽象层：不可再拆解的最小单元，其特点是责任单一，将车辆的传感器模块、执行器模块、ECU 等硬件抽象封装为设备抽象服务。此处以动力系统为例：温度传感器、水泵及风扇执行器等基本服务。

2) 原子层：即有具备独立的处理逻辑和仲裁的服务，也有组合其对应下层的设备抽象服务的服务集合，通过中间件标准接口为应用层服务提供按需编排的服务。以电器监控系统为例：可以封装车辆环境温度、电池电量、工作电流、工作电压等状态，针对热管理系统服务，通过设备抽象层的环境温度等数据，进行融合计算，控制水泵、压缩机、PTC 等协调工作；

3) 应用层：将原子层服务按照一定的时序重新编排组合，实现对整车应用的再定义，构建有差异化、有竞争力的应用软件，例如智能上电基于高压上电控制的原子服务，重新编辑高压上电的场景，生成一个新的应用软件。

服务接口调用需遵从如下原则实现对服务接口的需求关系管理和访问权限管理：

1) 上层服务接口逐层向下调用下层服务接口，不允许隔层调用，禁止下层服务接口反向调用上层服务接口；

2) 设备抽象层服务同层服务接口不可以相互调用。

服务接口调用原则图如图 5 所示。

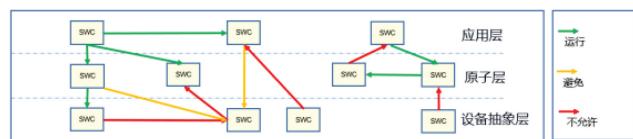


图 5 服务接口调用原则

2 中央式架构的关键技术

SOA 架构除采用了专门的车载以太网的物理层^[8]，SOA 在应用层还使用了两种不同的协议，分别是 2013 年纳入 AUTOSAR 4.1 的 SOME/IP 协议和 2018 年 3 月 AUTOSAR Adaptive 发布的 18-03 版平台标准材料中加入的 DDS 协议，这使得通信以面向服务的方式进行，下面简要对比说明一下两个协议的特点如图 6 所示。

对比项	SOME/IP	DDS
通信机制	Request/Response+订阅发布	订阅发布
架构风格	服务发现	全局数据空间
传输协议	TCP/UDP	缺省为 UDP，也支持 TCP
动态发现	是	是
Classic Autosar	支持	占用较大资源，一般不推荐
Adaptive Autosar	支持	支持
Linux/QNX	支持	支持
云端	不支持	需要 DDS Web 支持
安全性	TLS	自带安全协议，也支持 TLS 等

图 6 SOME/IP 与 DDS 协议对比图

我们在中央式架构设计实践中采用目前行业应用较多且较为成熟的 SOME/IP 协议，与 CAN 通信广播式的在总线上传输数据不同，SOME/IP 一种基于服务的通信方式，支持服务动态发布、动态订阅。SOME/IP 通信按照通信机制分为请求应答和提交订阅两类。其中请求应答类通信包括 Method、Getter、Setter；提交订阅类通信包括 Event 和 Field 中的 Notifier。SOMEIP 有 Method、Field、Event 三种通信调用类型其特点见表 1。

表 1 三种类型的服务接口特点

	主要用于同步调用，同步调用的返回参数有两种类型： 1) 通用类型，如不执行、将执行等。需要配合 Field-Getter 进行执行结果查询。 2) 直接返回参数结果。
RR Method	主要用于异步调用，不带返回结果，需要知道结果配合 Field 中的 Getter 获取。
FF method	请求消息中 payload 为空，响应消息的 payload 中具有该字段的值
Field-getter	请求消息的 payload 中有需要设置的值，响应消息的 payload 中有设置的值
Field-setter	按照订阅方需求发送状态 1) ChangeNotification，状态值发生变化时发送； 2) Cyclic，周期发送。
Field-notification	按照订阅方需求发送状态 1) ChangeNotification，状态值发生变化时发送； 2) Cyclic，周期发送； 3) 值超过一定阈值进行发送。
Event	按照订阅方需求发送状态 1) ChangeNotification，状态值发生变化时发送； 2) Cyclic，周期发送； 3) 值超过一定阈值进行发送。

3 SOA 的中央式架构开发实践

本章结合项目实践提出一种正向的基于 SOA 中央式架构开发方法，并结合一个功能场景基于该方法的设计实现例描述开发实践的过程，同时分析实践过程中存在的问题和给出的解决问题的方案。

3.1 一种正向的 SOA 中央式架构开发流程

本节结合项目实践描述一种从用户需求用例出发、以服务应用为向导的一种自顶向下的正向设计 SOA 中央式架构的开发方法，该开发方法总体流程图如图 7 所示。

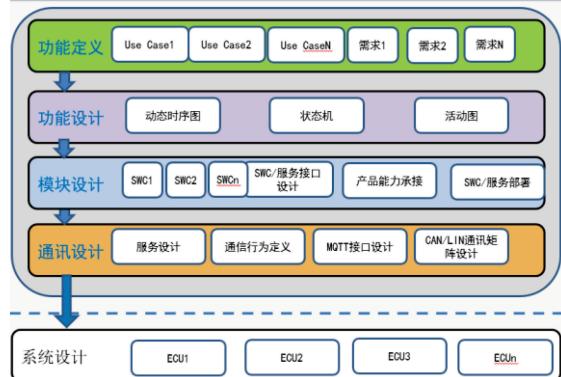


图 7 基于 SOA 架构的开发方法总体流程图

SOA 中央式架构开发流程主要步骤：

- 1) 功能定义。
- 2) 功能设计。
- 3) 模块设计。
- 4) 通信设计。
- 5) 系统设计。

3.1.1 功能定义

目的：基于整车的功能清单进行用例分析；通过用例分析功能流程、收集功能（人、机、法）需求；输出功能需求文档。

用例作为需求表达的一种标准方法，其在未知系统内部逻辑的情况下，实现表达功能完整流程和需求的特点，这与架构开发初期功能定义阶段输入仅有一份 Function List、功能实现内部策略未知的特点一致。功能定义阶段采用用例驱动的开发方法，分析 Function List 的业务需求和完整的功能流程，其特点是：

- 1) 产生功能的需求、提供功能业务可能性分析。
- 2) 为功能设计提供行为设计素材。
- 3) 实现标准化的语音翻译用户需求。

其中标准化语音要素为：参与者、前置条件、后置条件、触发条件、基本事件流、最主要的场景、备选事件流、异常事件流。

基于用例驱动的方法设计出功能用例（Use Case），采用标准化的语音翻译用户需求，实现了完整表达功能的流程，充分拆解分析对系统的需求，最终输出标准的功能需求文档。

3.1.2 功能设计

目的：用 UML 图将 Use Case 翻译为行为模型，将行为

的主体对象抽象并封装为候选产品能力（Product Capability, PC），完成功能分配的工作；在 UML 动态图（不限于序列图、活动图、状态机）中运用 PC 描述每个用例的功能实现链路；输出功能分配文档。

在功能设计阶段，通过以 UML 时序图为主，活动图、状态机为辅的图形化方式直观的描述 Use Case 的执行行为和功能实现逻辑以及链路。并根据中央架构将功能模型分为功能逻辑层和功能传感执行层的功能分配原则，将行为对象抽象并封装为候选的产品能力（PC）分配相应的层级模块，输出功能分配文档。

3.1.3 模块设计

目的：候选的产品能力（PC）经打合后形成产品能力库，并按层级分配到模块中；使用标准接口形式的服务软件单元描述模块内分配的产品能力，服务软件单元接口设计；进行服务软件单元的部署。

在模块设计阶段，候选产品能力打合后，形成固定的产品能力，经流程释放进入产品能力库，分配到模块中。根据功能分配信息，将服务软件单元要实现的产品能力，以 AUTOSAR 为标准，按照本文软件架构分层、服务接口颗粒度原则，设计服务软件单元的 SWC、Provide Interface、Require Interface、Interface Operation (Method/Field/Event)、Data Element。同时按照功能模型分层原则将服务软件单元部署到响应的 ECU。

3.1.4 通信设计

在完成服务软件单元到 ECU 的部署后可导出服务接口清单，按照 SOMEIP 通信标准完成服务接口的通信行为定义、SD (服务发现) 全局定义、SD ECU 定义；通过专用工具将服务接口矩阵生成汽车开放系统架构可扩展标记语言（Automotive Open System Architecture Extensible Markup Language, ARXML）文件，为应用软件代码开发输出通用的数据库文件，直接指导软件开发。

3.1.5 系统设计

在模块设计阶段完成服务接口的详细设计、服务部署到 ECU，在通信设计阶段完成了服务矩阵设计和 ARXML 文件制作，最终形成 ECU 软件系统需求规范，该文档用于软件开发团队或外部供应商进行 ECU 软件开发。

3.2 SOA 中央式架构开发实践

本节基于整车实施 SOA 中央式架构方案实际项目案例，以遥控钥匙解闭锁功能用例的架构开发与软件部署实践，结合实车数据、实车功能效果说明 SOA 中央式架构方案实施存在的一些问题和解决建议。

3.2.1 架构开发实践

遥控钥匙解闭锁功能用例按照上文描述的架构开发方法及设计原则，在原子服务层设计出钥匙控制、锁控制、喇叭控制、外部灯光控制服务，部署在中央计算平台，在设备抽象服务层设计出钥匙状态、锁状态、锁执行、喇叭状态、喇叭执行、危险警报灯状态、危险警报灯执行服务，部署在区域控制器，相关的服务划分、服务部署如图 8 所示。

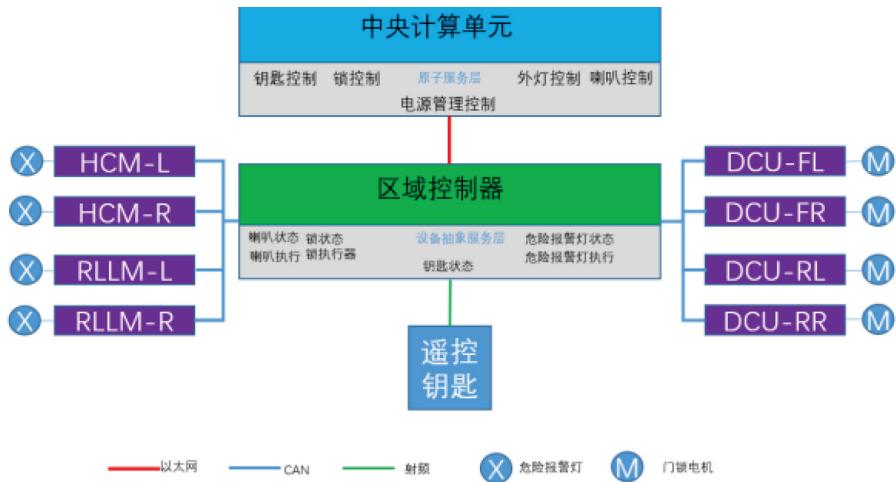


图 8 遥控钥匙解闭锁服务划分、服务部署图

3.2.2 功能实现链路与时序分析

SOA 中央式架构带来了功能的灵活组合与快速实施的优势，也导致了功能传递链路加长的问题。分别以车辆休眠、

车辆上电状态分析遥控钥匙解闭锁过程链路与响应时序以及产生的功能影响。遥控钥匙解闭锁过程链路与响应时序图如图 9、图 10 所示：

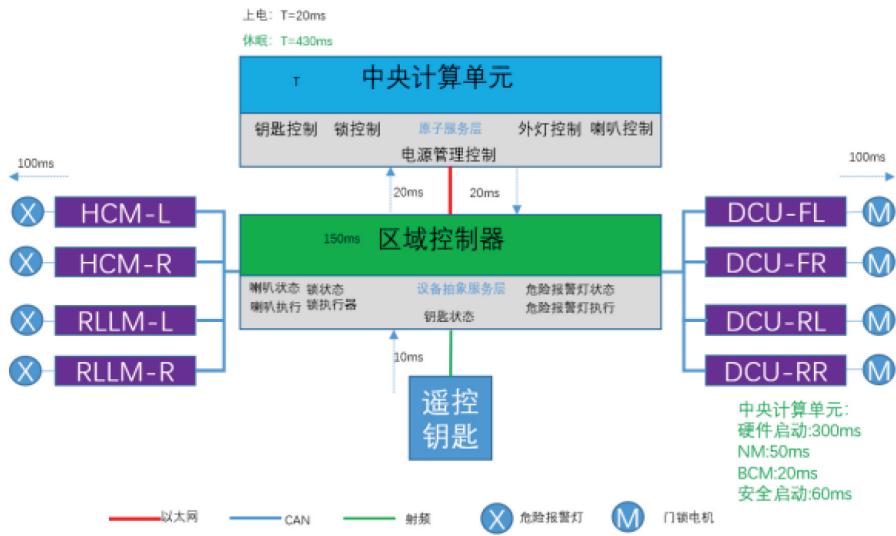


图 9 遥控钥匙解闭锁链路图

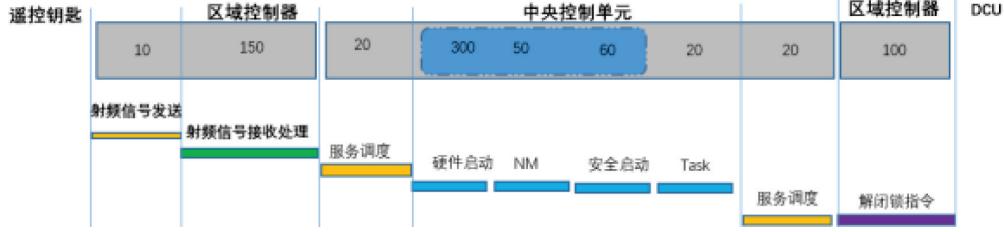


图 10 遥控钥匙解闭锁响应时序图

从图 10 可以看出，遥控钥匙解闭锁链路一致为：钥匙模块射频发送解锁信号→区域控制器反馈钥匙状态→中央计算单元钥匙控制处理钥匙逻辑→中央计算单元锁控制处理解锁逻辑→区域控制器执行解锁指令→DCU 驱动锁电机解锁。遥控钥匙解闭锁响应时序二者存在着 410ms 的差异，产生的原因是在于车辆休眠模式下，由于中央控制单元硬件启动耗时 300ms、网络管理模块耗时 50ms、安全启动模块耗时

60ms 的叠加影响造成的。

3.2.3 功能的影响

在车辆唤醒模式下，从遥控钥匙解锁按键按下到锁模块执行解锁耗时约 200ms；在休眠模式下，由于中央控制单元需要被区域控制器唤醒并完成初始化需多用时 410ms。基于中央式架构模式，逻辑处理单元部署在中央控制单元中，钥匙的按键状态信息被区域控制器接收后直接传到中央控制单

元，由中央控制单元对钥匙状态进行逻辑处理，最终识别钥匙的指令意识，当用户的遥控钥匙按键时间短于中央控制器

单元的初始化时间，会导致钥匙状态信息丢失，休眠模式下偶发遥控钥匙指令不响应的车辆实物问题数据如图 11 所示：

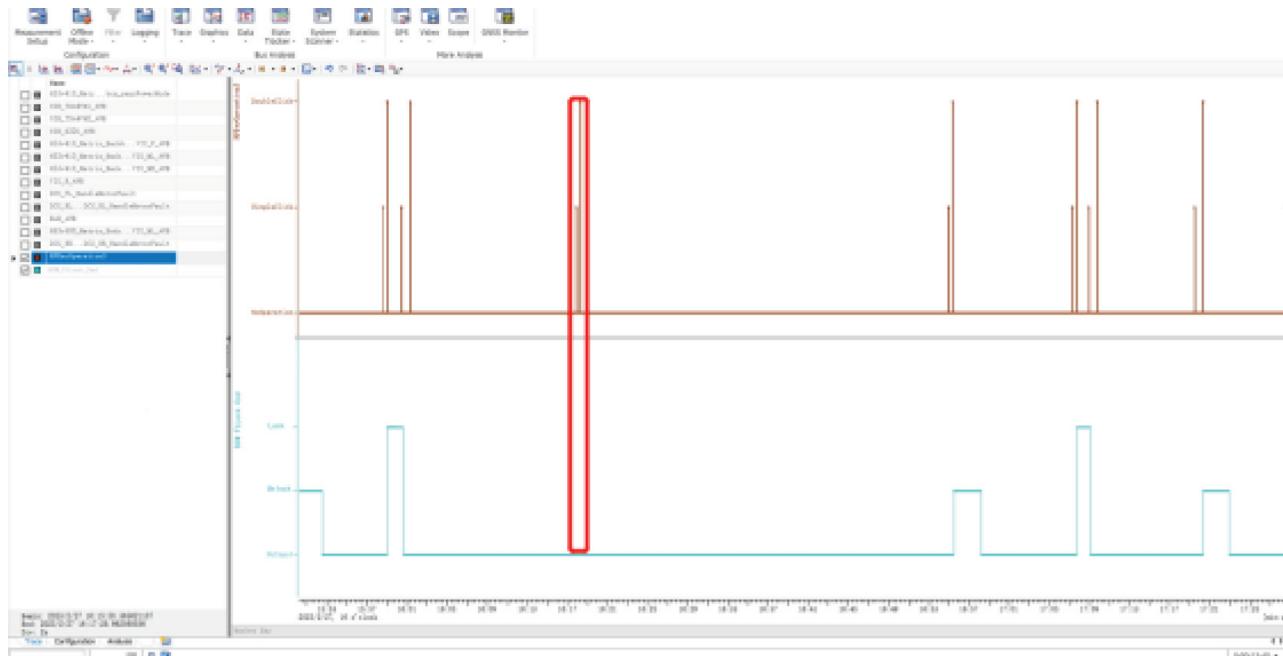


图 11 休眠模式下偶发遥控钥匙指令不响应

3.2.4 问题总结与对策建议

遥控钥匙解锁偶发失效总结起来是由于如下原因造成：功能处理逻辑分层后功能链路加长；休眠模式中央控制器单元初始化时间长；区域控制器对于本地就近连接的传感器信号无处理逻辑过度依赖中央控制单元。与钥匙还有类似问题的有脚踢传感器被误触发时导致误唤醒整车引起车辆馈电问题；低压蓄电池智能传感器在蓄电池低电量时持续唤醒车辆引起车辆馈电问题。

对策建议：根据实施案例中实车的用户体验与表现，我们认为并非所有逻辑处理都适合部署到中央控制单元上，在遵循 SOA 中央式架构开发流程与原则的前提下，需结合功能实际分析场景与时序特点，调整功能逻辑部署，给出如下建议：

1) 可能导致整车异常休眠唤醒的唤醒源的处理逻辑，例如：脚踢信号处理逻辑、低压蓄电池智能传感器智能补电逻辑；

2) 车辆进入功能的唤醒信号处理逻辑，例如：遥控钥匙控制车辆、开启尾门功能等逻辑；

3) 控制器逻辑与硬件时序强关联的逻辑，例如：锁释放锁、锁吸合与电机驱动控制逻辑。

4) 带有复杂逻辑无法集成的电控系统处理逻辑，例如：底盘的车辆稳定性控制模块、动力的发动机模块等。

4 结论

正如汽车朝着电动化、网联化、智能化的趋势不可阻挡一样，汽车电子电器架构向中央式架构发展是大势所趋，文章从中央式架构的特点、关键技术、正向流程方法、描述了 SOA 的中央式架构开发方法，总结了 SOA 架构开发落地实践过程中存在的挑战，并给出了解决问题的方法。SOA 中央式架构开发在行业里没有成熟的落地经验可以借鉴，需要在实践总不断总结不断优化，发挥其优势解决其短板。

参 考 文 献

- [1] KUGELE S, OBERGELL P, BROY M, et al. On Service—Orientation for Automotive Software [A] IEEE. 2017 IEEE International Conference on Software Architecture (C). Gothenburg: IEEE, 2017.
- [2] 刘佳熙, 施思明, 徐振敏, 等. 面向服务架构汽车软件开发方法和实践 [J]. 中国集成电路, 2021 (Z1).
- [3] David Sprott, Lawrence Wilkes. Understanding Service-Oriented Architecture [M]. [S. l. : s. n.], 2004.
- [4] Donovan Porter. 100BASE-T1 Ethernet: the evolution of automotive networking. Technical Report [M]. [S. l. : s. n.], 2018.
- [5] Kay Weckemann. Domanenübergreifende Anwendungskommunikation im IP-basierten Fahrzeugbordnetz [D]. Ludwig-Munchen: MaximiliansUniversitat, 2014.