车控操作系统总体技术要求研究报告

汽标委智能网联汽车分标委

资源管理与信息服务标准工作组

2021 年 7 月

目　录

前　言

智能网联汽车是解决汽车社会交通安全、道路拥堵、能源消耗、污染排放等问题的重要手段，也是构建智慧出行服务产业生态的核心要素和推进交通强国、数字中国、智慧城市的重要载体。

车载智能计算基础平台是智能网联汽车核心新型增量零部件。车控操作系统是智能网联汽车的基础软件部分，运行于智能网联汽车车载智能计算基础平台。智能网联汽车的复杂性，需要通过车控操作系统软件架构和接口的标准化实现产业链的分工协同，提高开发效率，保障软件平台的安全可信，减少对接成本，构建统一生态。

在此衷心感谢参加研究报告编写的各单位、组织及个人。

组织指导：汽标委智能网联汽车分标委

牵头单位：国汽(北京)智能网联汽车研究院有限公司、华为技术有限公司

参与单位：国汽智控（北京）科技有限公司、中国汽车技术研究中心有限公司、北京地平线机器人技术研发有限公司、电子科技大学、中国第一汽车股份有限公司、一汽解放汽车有限公司、上汽大众汽车有限公司、北汽福田汽车股份有限公司、惠州市德赛西威汽车电子股份有限公司、上汽通用五菱汽车股份有限公司、长城汽车股份有限公司、东风汽车有限公司东风日产乘用车公司、上海汽车集团股份有限公司零束软件分公司、大众汽车（中国）投资有限公司、中汽研软件测评（天津）有限公司、北京百度智行科技有限公司、江淮汽车集团股份有限公司、长安汽车软件科技有限公司、清华大学、东风汽车集

团有限公司技术中心、东风商用车有限公司、国家 ITS 中心智能驾驶研究院、北京汽车股份有限公司、阿里巴巴（中国）有限公司、高通无线通信技术(中国)有限公司、中国汽车工程研究院股份有限公司、中兴通讯股份有限公司、中国软件评测中心（工业和信息化部软件与集成电路促进中心)、上海机动车检测认证技术研究中心有限公司、斑马网络科技有限公司、上海瓶钵信息科技有限公司

参与人员：田思波、丛炜、潘晏涛、褚文博、周铮、程智锋、吴含冰、罗蕾、石庆鹏、张晓谦、孔涛、孟宪刚、郑岩、石景文、余得水、钱国平、伍宇志、罗覃月、周思儒、邹德英、孙华、郑四发、樊胜利、吴丹丹、唐波、周明珂、彭琪惠、徐耀宗、彭伟、冯锦文、吴琼、朱乾勇、黄懿、桂绍靖、江浩、刘珺、王琳、刘红军、李俨、凌晓峰、徐立锋、郭盈、李玉坷、高长胜、金一华、刘时珍、王锦、张路、刘晨曦、程唐平、田俊涛、马涛、王圭、许梦枚、林智桂、王萌、金天、侯昕田、丁钊、李毓强、郑方、殷苏辰、陈书平、贾元辉、秦文婷、刘大鹏、崔云峰、周波、刘海威、满志勇

1 术语定义及缩略语

1.1 术语与定义

下列术语与定义适用于本文件。

1.1.1 车载智能计算基础平台 intelligent vehicle base computing platform

支撑智能网联汽车驾驶自动化功能实现的软硬件一体化平台，包括芯片、模组、接口等硬件以及系统软件、功能软件等软件，以适应传统电子控制单元向异构高性能处理器转变的趋势。

1.1.2 车用操作系统 vehicle operating system

运行于车内的系统程序集合，以实现管理硬件资源、隐藏内部逻辑提供软件平台、提供用户程序与系统交互接口、为上层应用提供基础服务等功能，包含车控操作系统和车载操作系统。

1.1.3 车控操作系统 vehicle-controlled operating system

运行于车载智能计算基础平台异构硬件之上，支撑智能网联汽车驾驶自动化功能实现和安全可靠运行的软件集合。

1.1.4 系统软件 system software

车控操作系统中支撑驾驶自动化功能实现的复杂大规模嵌入式系统运行环境，分为安全车控系统软件和智能驾驶系统软件。

1.1.5 功能软件 function software

车控操作系统中根据面向服务的架构设计理念，通过提取智能驾驶核心共性需求，形成智能驾驶各共性服务功能模块，高效实现驾驶自动化功能开发的软件模块。

1.1.6 域控制器 domain controller unit

根据汽车电子部件功能将整车划分为动力总成、智能座舱和智能驾驶等几个域，集中控制域内原本归属各个 ECU 的大部分功能，以取代传统的分布式架构。

1.1.7 驾驶自动化功能 autonomous driving automation feature

驾驶自动化系统在特定的设计运行范围内执行动态驾驶任务的能力。

1.1.8 实时安全域 realtime safety domain

车控操作系统中对应安全车控操作系统的系统软件部分，主要有实时操作系统、硬件抽象层、基础软件、运行时环境和实时域功能服务组成。通过其上运行的各种检测任务针对获取到的各种数据进行监测判断以确定当前车辆是否处于安全状态，并在必要的时候，通过其上运行的安全控制应用实现对车辆的安全控制。

1.1.9 云控技术 cloud integration control technology, CICT

车路云一体化融合控制技术的简称，是车路云一体化融合控制系统通过车路融合感知、融合决策与控制，实现车辆行驶和交通运行安全与效率综合提升的技术，以下简称“云控”。

2.2 缩略语

下列缩略语适用于本文件。

ADAS 高级驾驶辅助系统 Advanced Driving Assistance System

AI 人工智能 Artificial Intelligence

API 应用程序编程接口 Application Programming Interface

ARA AUTOSAR 自适应应用运行环境 AUTOSAR Runtime for Adaptive Applications

ARP 地址解析协议 Address Resolution Protocol

ASIL 汽车安全集成等级 Automotive Safety Integration Level

AUTOSAR 汽车开放系统架构 AUTomotive Open System Architecture

AVB 以太网音视频桥接技术 Ethernet Audio/Video Bridging

CA 条件自动驾驶 Conditional Driving Automation

CANBus 控制器局域网总线 Controller Area Network Bus

CNN 卷积神经网络 Convolutional Neural Network

CPU 中央处理器 Central Processing Unit

DDS 数据分发服务 Data Distribution Service

DoS 拒绝服务 Denial of Service

DSP 数字信号处理 Digital Signal Process

ECU 电子控制单元 Electronic Control Unit

EEA 电子电气架构 Electrical/Electronic Architecture

FA 全自动驾驶 Full Driving Automation

FIFO 先进先出 First Input First Output

EMC 电磁兼容 Electro Magnetic Compatibility

FPGA 现场可编程逻辑门阵列 Field Programmable Gate Array

GPL GNU 通用公共许可证 GNU General Public License

GPU 图形处理器 Graphics Processing Unit

HA 高级自动驾驶 High Driving Automation

HMI 人机接口 Human Machine Interface

I2C 两线式串行总线 Inter-Integrated Circuit

I2S 集成电路内置音频总线 Inter-IC Sound

IDE 集成开发环境 Integrated Development Environment

IDL 接口定义语言 Interface Definition Language

IPC 工业控制计算机 Industrial PC

JasPar 日本汽车软件平台架构组织 Japan Automotive Software Platform Architecture

MCU 微控制单元 Microcontroller Unit

OBD 车载自动诊断系统 On Board Diagnostics

OEM 原始设备制造商 Original Equipment Manufacturer

OSEK 汽车电子开放式系统及接口规范 Open Systems and the corresponding interfaces for automotive Electronics

OTA 空中下载 Over the Air

POSIX 可移植操作系统接口 Portable Operating System Interface of UNIX

QoS 服务质量 Quality of Service

VDX 汽车分布式执行标准 Vehicle Distributed Executive

RAS 可靠性、可访问性和可服务性 Reliability、Accessibility& Serviceability

RGMII 精简吉比特介质独立接口 Reduced Gigabit Media Independent Interface

ROS 机器人操作系统 Robot Operating System

RTOS 实时操作系统 Real Time Operating System

SLAM 同步定位与建图 Simultaneous Localization and Mapping

SOA 面向服务的架构 Service-Oriented Architecture

SPI 串行外设接口 Serial Peripheral Interface

SSD 固态驱动器 Solid State Disk

TEE 可信执行环境 Trusted Execution Environment

TLS 安全传输层协议 Transport Layer Security

TSN 时间敏感网络 Time Sensitive Network

UART 通用异步收发传输器 Universal Asynchronous Receiver/Transmitter

USB 通用串行总线 Universal Serial Bus

V2X 车联网通信 Vehicle to Everything

VSLAM 视觉同步定位与地图构建 Visual SLAM

2 车控操作系统研究背景

汽车产业是国民经济的重要支柱产业，带动庞大的制造业上下游，代表国家工业水平，是“制造强国”重大战略部署的重要支撑和融合载体。当前，世界汽车产业正在经历一场以“电动化、智能化、网联化和共享化”为特征的“新四化”技术革命和行业变革，随着新一代能源技术变革、信息技术、人工智能、大数据等与汽车“新四化”的加速融合，以汽车为代表的运载工具正在成为全球技术变革和科技创新的竞争制高点。发展智能网联汽车是解决汽车社会交通安全、道路

拥堵、能源消耗、污染排放等问题的重要手段，也是构建智慧出行服务产业生态的核心要素和推进交通强国、数字中国、智慧城市的重要载体。

2020 年 11 月国务院印发《新能源汽车产业发展规划(2021－2035 年)》部署了 5 项战略任务，其中构建新型产业生态作为其中一项，鼓励以生态主导型企业为龙头，加快车用操作系统开发应用。同年发布的《智能网联汽车技术路线图 2.0》支撑构建了中国智能网联汽车产业技术发展体系，为产业技术发展指明了方向，对于中国智能网联汽车产业动态与趋势，出现的技术新特征与趋势进行研判。

车载智能计算基础平台是智能网联汽车最核心的新型增量零部件,是智能网联汽车产业互联网下的最基础平台，是兼顾市场亟需与国家监管的“软硬”一体高科技产品。2019 年 5 月，在工信部指导下，赛迪研究院、装备中心和国汽智联等联合业内优势单位发布《车载智能计算基础平台参考架构 1.0》白皮书，涵盖网联、云控、数据通信、地图、信息安全等中国特色，初步形成车控操作系统中国共识。车控操作系统是运行在异构分布硬件架构上的实时安全平台软件，提供整车及部件感知、规划、控制等功能框架并向上支撑智能网联驾驶生态的软件集合，是车载智能计算基础平台安全、实时和高效运行的重要基础和核心支撑。

2019 年 10 月，汽标委发布了《车用操作系统标准体系》，规范了车用操作系统定义，划分了车用操作系统边界，明确了车用操作系统分类（如下图 1 所示），构建了车用操作系统标准体系。《车用操作系

统标准体系》的发布，为车控操作系统标准化工作的开展提供了指导方向。

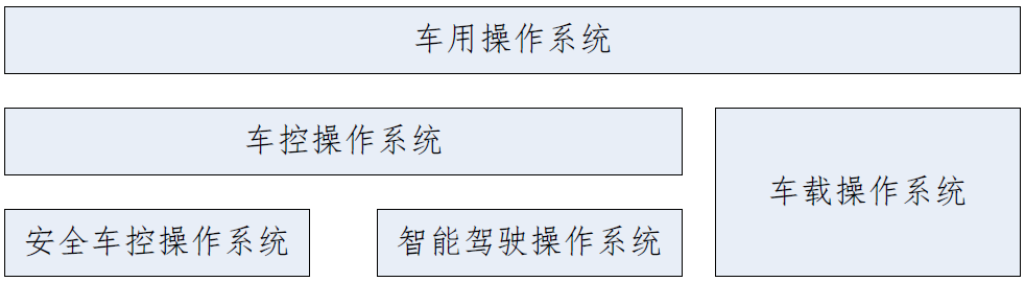

图 1 车用操作系统分类

车控操作系统分为安全车控操作系统和智能驾驶操作系统，其中，安全车控操作系统主要面向经典车辆控制领域，如动力系统、底盘系统和车身系统等，该类操作系统对实时性和安全性要求极高，生态发展已趋于成熟。智能驾驶操作系统主要面向智能驾驶领域，应用于智能驾驶域控制器，该类操作系统对安全性和可靠性要求较高，同时对性能和运算能力的要求也较高。该类操作系统目前在全世界范围内都处于研究发展的初期，生态尚未完备。

车载操作系统主要面向信息娱乐和智能座舱，主要应用于车机中控系统，对于安全性和可靠性的要求处于中等水平，该类操作系统发展迅速，依托于该类操作系统的生态也处于迅速发展时期。

本研究报告的涉及范围主要是车控操作系统。

2.1 车控操作系统需求分析

汽车电子电气架构（EEA，Electrical/Electronic Architecture）是集合了汽车的电子电气系统原理设计、中央电器盒设计、连接器设计、电子电气分配系统等设计为一体的整车电子电气解决方案。汽车电子电气架构具体是在功能需求、设计要求和标准法规等特定约束下，通

过对功能、性能、成本和装配等各方面进行分析，将动力总成、智能驾驶系统、信息娱乐系统等信息转化为实际的电源分配物理布局、信号网络、数据网络、诊断、电源管理等电子电气解决方案。

随着汽车智能化、网联化趋势的发展，汽车电子底层硬件不再是由单一芯片提供简单的逻辑计算，而是需要复杂的多核芯片提供更为复杂控制逻辑以及强大的算力支持。软件也不再是基于某一固定硬件开发，而是要具备可移植、可迭代和可扩展等特性。汽车智能化与网联化发展趋势共同推动了汽车电子电气架构的变革，一方面是车内网络拓扑的优化和实时、高速网络的启用，另一方面是电子控制单元（ECU）的功能进一步集成到域控制器甚至车载计算机。汽车电子电气架构的演进趋势为从分布式架构到域集中式架构最后到中央集中式架构转变，如图 2 所示。

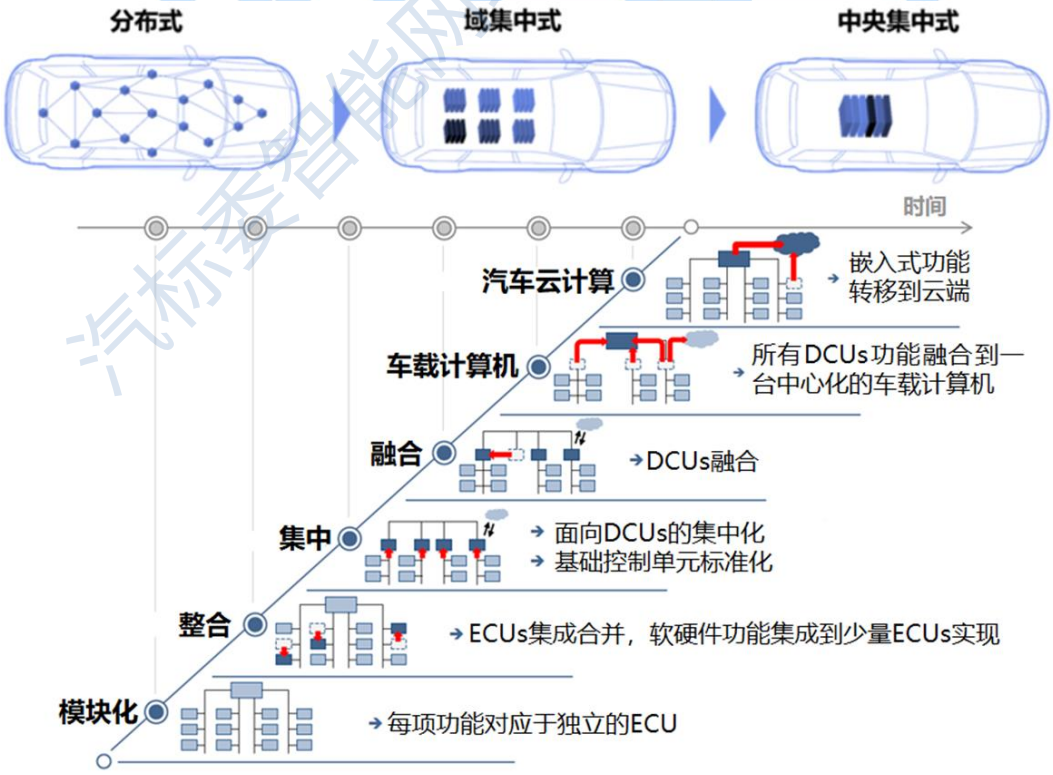

图 2 汽车电子电气架构演进趋势（来源于博世）

当前，集中式电子电气架构的主要类型有三域 EEA（如图 3 所示）和 Zonal EEA（如图 4 所示）。本研究报告主要是针对三域 EEA 架构类型。其中，三域主要是指车辆控制域、智能驾驶域和智能座舱域，车辆控制域是将原动力域、底盘域和车身域等经典车辆域进行了整合，负责整车控制，实时性及安全性要求高；智能驾驶域负责自动驾驶相关感知、规划、决策相关功能的实现；智能座舱域负责 HMI 交互和智能座舱相关功能的实现。对于集中式电子电气架构，智能驾驶系统功能的实现在其中扮演重要地位，相对于分布式电子电气架构，集中式电子电气架构可以提供更为强大的算力，而强大的算力实现不仅需要硬件的支持，更需要和集中式电子电气架构适配的先进车控操作系统。随着软件定义汽车、数字驱动设计趋势的发展，车控操作系统作为车载智能计算基础平台的核心技术，也将成为未来智能汽车领域竞争的重要方向。

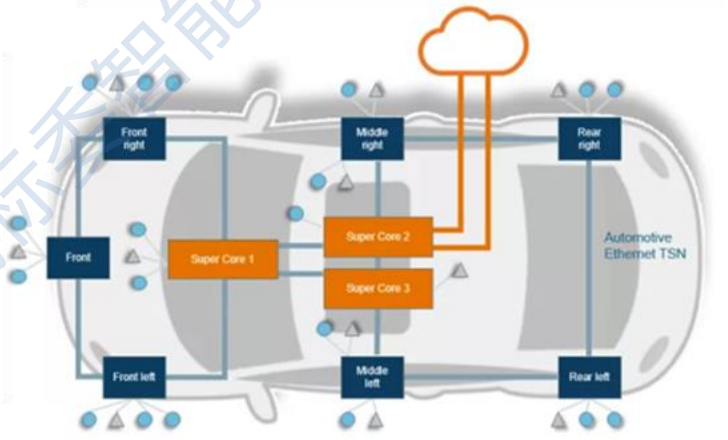

图 3 三域 EEA 示意图（来源于网络）

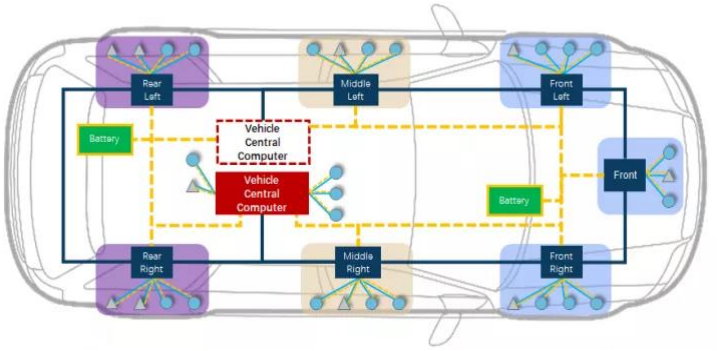

图 4 Zonal EEA 示意图（来源于网络）

2020 年 10 月，《节能与新能源汽车技术路线图 2.0》发布，其中提出了智能计算平台的技术发展路线（参考表 1），进一步从战略层面定位了计算平台及其操作系统在智能驾驶汽车发展中的重要作用。

表 1 计算平台技术发展路线

2025	2030	2035
计算平台支持 CA 级别自动驾驶。硬件平台实现核心模块的集成，系统软件实现全栈、完整化构建能力，功能软件实现框架完整构建能力，自动驾驶操作系统实现自主知识产权的突破，初步建立自主开发生态。		
计算平台支持全面网联和云控协同感知、决策、控制，数据收集与处理，支持 HA、FA 级自动驾驶。硬件平台实现计算平台核心模块异构分布架构，系统软件主要部分实现自主，功能软件在部分重要模块的实现与应用基础上实现自主可控，在自主可控能力之上建立功能软件级生态。		
计算平台具备和交通基础设施（车路云）全方位无缝协同的能力。架构方面实现车辆控制单元高度集成及异构融合方案，算法上支持强人工智能。硬件平台在芯片级集成基础上实现完全自主知识产权，系统软件与功能软件实现全面自主化，引领全行业的标准。计算平台实现定制与行业领先，建立自主可控的开发与应用生态。		

车控操作系统包括安全车控操作系统和智能驾驶操作系统。安全车控操作系统主要是实时操作系统（RTOS），主要应用对象是 ECU。ECU 对安全车控操作系统最基本的要求是高实时性，系统需要在规定时间内完成资源分配、任务同步等指定动作。嵌入式实时操作系统

具有高可靠性、实时性、交互性以及多路性的优势，系统响应极高，通常在毫秒或者微秒级别，满足了高实时性的要求。目前，主流的安全车控操作系统都兼容 OSEK/VDX 和 Classic AUTOSAR 这两类汽车电子软件标准。其中，Classic 平台基于 OSEK/VDX 标准，定义了安全车控操作系统的技术规范。

随着智能化、网联化技术的发展，智能汽车感知融合、决策规划和控制执行功能带来了更为复杂算法并产生大量的数据，需要更高的计算能力与数据通讯能力。基于 OSEK/VDX 和 Classic AUTOSAR 软件架构的安全车控操作系统已经不能满足未来自动驾驶汽车的发展需求，AUTOSAR 组织为面向更复杂的域控制器和中央计算平台的集中式电子电气架构推出 Adaptive AUTOSAR 平台。Adaptive 平台定义采用了基于 POSIX 标准的操作系统，可以为支持 POSIX 标准的操作系统及不同的应用需求提供标准化的平台接口和应用服务，主要是为了适应汽车智能化的发展需求。Adaptive AUTOSAR 尚处于发展初期，其生态建设获得 Tier1、主机厂的普遍认可尚需时日。同时，由于智能网联汽车的区域属性及社会属性增加，在行驶过程中需要通信、地图、数据平台等本国属性的支撑和安全管理，每个国家都有自己的使用标准规范，因此，Adaptive AUTOSAR 能否满足智能化的需求尚有待验证。

智能汽车的发展需要符合中国标准的车内电子电气架构、通信系统、智能终端、驾驶辅助和自动驾驶系统、云平台等新架构汽车产品标准，因此，在软件定义汽车的趋势下，网联、云控也成为中国智能

网联汽车发展的特色，而作为智能网联汽车的核心车控操作系统，更应该满足“中国标准”方案的智能汽车电子电气架构。此外，车控操作系统的架构设计某种程度上决定能否建立良好的应用软件生态，从而建立基于软件、服务的商业模式。

综上所述，鉴于适配于集中式电子电气架构的车控操作系统软件架构尚在演化，以及“中国标准”方案的智能网联汽车的发展需求，研究并制定符合“中国方案”智能网联汽车发展的车控操作系统架构具有迫切需求和重要意义。

2.2 车控操作系统研究现状

2.2.1 安全车控操作系统

安全车控操作系统国外发展较早，目前已经开展了一系列的标准化工作，国内目前主要处于跟随状态。

欧洲在 20 世纪 90 年代发展出用于汽车电子上分布式实时控制系统的开放式系统标准 OSEK/VDX，主要包括 4 部分标准：1）操作系统规范（OS）；2）通信规范；3）网络管理规范；4）OSEK 实现语言。但随着技术、产品、客户需求等的升级，OSEK 标准逐渐不能支持新的硬件平台。

2003 年，宝马、博世、大陆、戴姆勒、通用、福特、标志雪铁龙、丰田、大众 9 家企业作为核心成员，成立了一个汽车开放系统架构组织（简称 AUTOSAR 组织），致力于建立一个标准化平台，独立于硬件的分层软件架构，制定各种车辆应用接口规范和集成标准，为应用开发提供方法论层面的指导，以减少汽车软件设计的复杂度，提高汽

车软件的灵活性和开发效率，以及在不同汽车平台的复用性。AUTOSAR 以 OSEK/VDX 为基础，但涉及的范围更广。

截至目前，AUTOSAR 组织已发布 Classic 和 Adaptive 两个平台规范，分别对应安全控制类和自动驾驶的高性能类。Classic 平台基于 OSEK/VDX 标准，定义了安全车控操作系统的技术规范。Classic AUTOSAR 的软件架构如图 5 所示，其主要特点是面向功能的架构（FOA），采用分层设计，实现应用层、基础软件层和硬件层的解耦。

图 5 Classic AUTOSAR 软件架构

AUTOSAR 标准平台由于采用开放式架构和纵向分层、横向模块化架构，不仅提高了开发效率，降低开发成本，同时保障了车辆的安全性与一致性。AUTOSAR 组织发展至今，得到了越来越多的行业认可，目前已有超过 180 家的车、零部件、软件、电子等领域的成员。AUTOSAR 目前已经成为国际主流的标准软件架构，基于 AUTOSAR 标准平台，拥有完整的汽车软件解决方案的企业主要有 Vector、KPIT、

ETAS、DS 以及被大陆收购的 Elektrobit 和被西门子收购的 MentorGraphics。此外，宝马、沃尔沃等汽车厂商都相继推出了基于 AUTOSAR 标准平台的车型。

在日本，日本汽车软件平台架构组织（Japan Automotive Software Platform Architecture，JasPar）成立于 2004 年，旨在联合企业横向定制兼顾汽车软硬件的通信标准、实现车控操作系统的通用化，提高基础软件的再利用率等。JasPar 组织成员包括绝大多数的日系汽车及配套软硬件产品厂商。

我国主机厂及零配件供应商目前主要使用 Classic AUTOSAR 标准进行软件开发。一汽集团、长安集团等主机厂于 2009 年开始利用 Classic AUTOSAR 标准的工具进行 ECU 的设计、开发、验证。同时，上汽集团、一汽集团、长安集团、奇瑞集团等主机厂和部分高校成立了 CASA 联盟，旨在中国推广和发展 AUTOSAR 架构。目前江淮汽车也是主要基于 Classic AUTOSAR 标准进行软件和产品开发。

在产品方面，普华软件是中国电子科技集团的国产操作系统战略平台，并作为牵头单位承担了关于汽车电子操作系统的十一五、十二五核高基重大专项，所形成的车控操作系统在车身控制模块（BCM）、新能源整车控制器（VCU/HCU）、电子转向系统（EPS）等关键零部件得到量产应用，并已被德国博世的先进辅助驾驶系统（ADAS）量产使用。

东软睿驰发布了 NeuSAR 产品，其基于 AUTOSAR 研发制作，为自主研发自动驾驶系统的 OEM 整车企业及零部件供应商提供的面

向下一代汽车通讯和计算架构的系统平台，包含 AUTOSAR Classic、AUTOSAR Adaptive 及系列开发系统工具。

2.2.2 智能驾驶操作系统

智能驾驶操作系统将会成为自动驾驶汽车发展的核心竞争力之一，由于安全车控操作系统相对成熟，且智能驾驶操作系统部分包含安全车控操作系统，所以本文提到的车控操作系统主要是指智能驾驶操作系统。智能驾驶操作系统发展趋势和特点是纵向分层，以实现层与层之间的解耦，方便快速开发和移植，如图 6 所示。

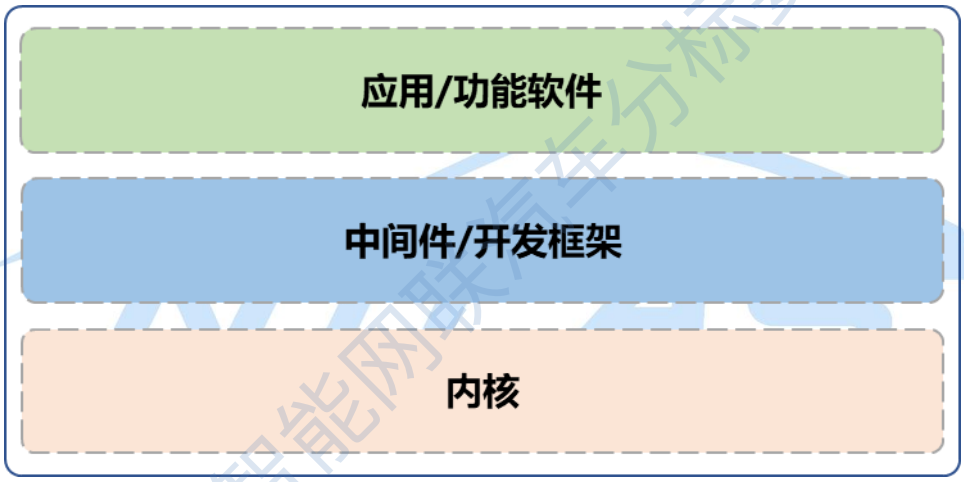

图 6 智能驾驶操作系统纵向分层示意图

AUTOSAR 组织为应对自动驾驶技术的发展推出了 Adaptive AUTOSAR（AP）架构，如图 7 所示，其主要特点是采用面向服务的架构（SOA），服务可根据应用需求动态加载，可通过配置文件动态加载配置，并可进行单独更新，相对于 Classic AUTOSAR（CP），可以满足更强大的算力需求，更安全，兼容性好，可进行敏捷开发。

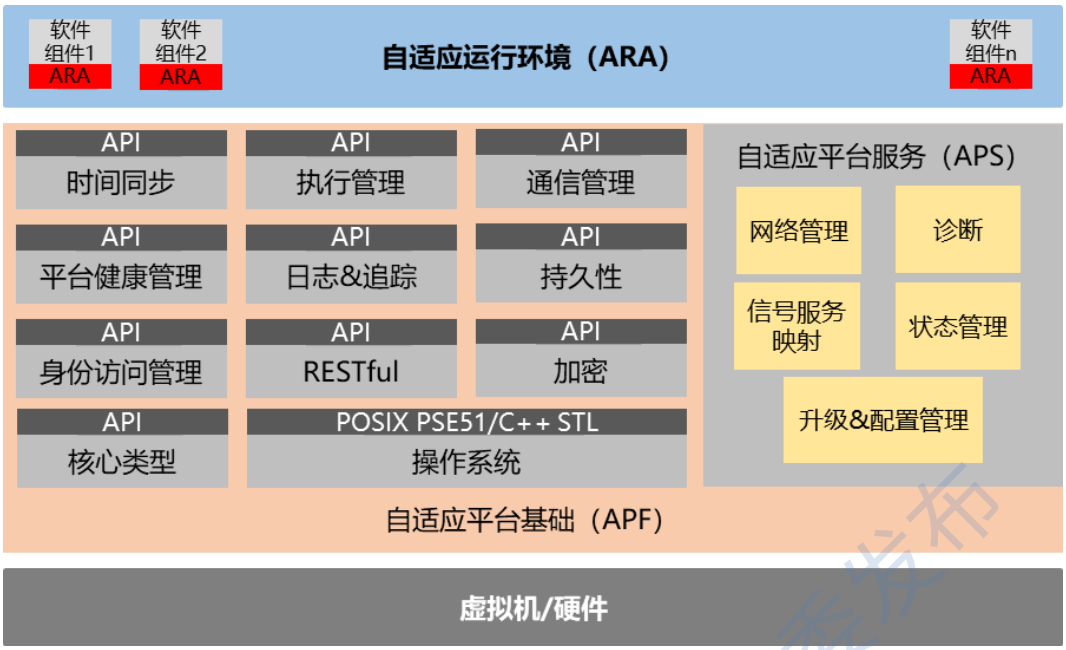

图 7 Adaptive AUTOSAR 架构

Adaptive AUTOSAR 系统主要适应于新的集中式的高性能计算平台，满足车内部件之间的高速通信需求和智能驾驶的高计算能力需求。AP 平台采用了服务化的架构，系统由一系列的服务组成，应用和其他软件模块可以根据需求调用其中的一个或者多个服务，而服务可以是平台提供的，也可以是远程其他部件提供，OEM 可以按照功能设计需求定义自己的服务组合。AP 设计也在 CP 成功的设计方法论的指导下进行开发，兼顾灵活和功能安全的确定性需求，同时可以进行整车功能的统一设计，兼顾安全车控和智能驾驶系统。AP 平台没有设计新的操作系统内核，所有符合 POSIX PSE51 接口的操作系统内核都可以使用，AP 平台重点是在操作系统内核之上的系统服务中间层，主要分为平台基础功能和平台服务功能两部分。平台基础功能更多的是本地服务，采用库的方式进行调用，如生命周期管理、通信管理、存储管理、诊断功能等；平台服务不限制服务的提供实例的

位置，采用互联网常用的服务类似的调用方式，如升级管理、网络管理、状态管理、传感器服务接口等。AP 平台主要的三个支撑和演进方向是：安全（包含信息安全和功能安全），连接（包括车内和车外各种新的通信机制），可升级（包含 OTA，灵活的软件设计和管理等）。AP 平台仍采用传统的标准设计方式，每年一个版本集中进行新的功能发布。

虽然 AUTOSAR AP 在继承 ATUOSAR CP 成功经验的基础上，采用了很多互联网的思路来设计，但是由于汽车软件需求变化较快，仍存在很多问题，如传统的 AUTOSAR 方法论配置繁琐，支持的通信速率低，不支持解耦部署策略和动态升级方案，特别是车车协同、车路协同、车云协同等还一直处于需求讨论阶段，没有支持的时间表。AUTOSAR AP 平台是适应新一代电子电气架构下的集中式计算需求而产生的，但只是整车功能中的一小部分，缺乏从整车电子电气系统视角考虑信息安全、功能安全、通信等需求。

在底层操作系统之上，软件中间件在智能驾驶领域也备受关注。中间件的主要目标是为上层应用提供数据通信、协议对齐、计算调度、模块化封装等常用功能，为应用开发提供标准化、模块化的开发框架，实现模块解耦和代码复用。ROS 作为最早开源的机器人软件中间件，很早就被机器人行业使用，很多知名的机器人开源库，比如基于 quaternion 的坐标转换、3D 点云处理驱动、定位算法 SLAM 等都是开源贡献者基于 ROS 开发的。ROS 的首要设计目标是在机器人研发领域提高代码复用率。ROS 是一个分布式的进程（也就是“节点”）

框架，这些进程被封装在易于被分享和发布的程序包和功能包中。整个智能驾驶系统和机器人系统有很强的相似度，ROS 的开源特性，丰富的开源库和工具链，特别在智能驾驶的研究领域有着较为广泛的应用，很多自动驾驶的原型系统中都能够看到 ROS 的身影，例如 AUTOWARE，百度 Apollo 最初也是使用了 ROS，直至 Apollo 3.5 版本才切换至自研的车载中间件 Cyber RT。

ROS 在发展过程中主要有两个版本，ROS1 和 ROS2，ROS1 的通信依赖中心节点的处理，无法解决单点失败等可靠性问题。为了更好的符合工业级的运行标准，ROS2 最大的改变是，取消 Master 中央节点，实现节点的分布式发现，发布/订阅，请求/响应；底层基于 DDS（数据分发服务）这个工业级的通信中间件通信机制，支持多操作系统，包括 Linux、windows、Mac、RTOS 等。

虽然 ROS2 基于 ROS1 有了很大的改进，但是距离完全车规应用还有很大的距离，有些公司如 APEX.AI 也在对 ROS 进行车规级的改造尝试。

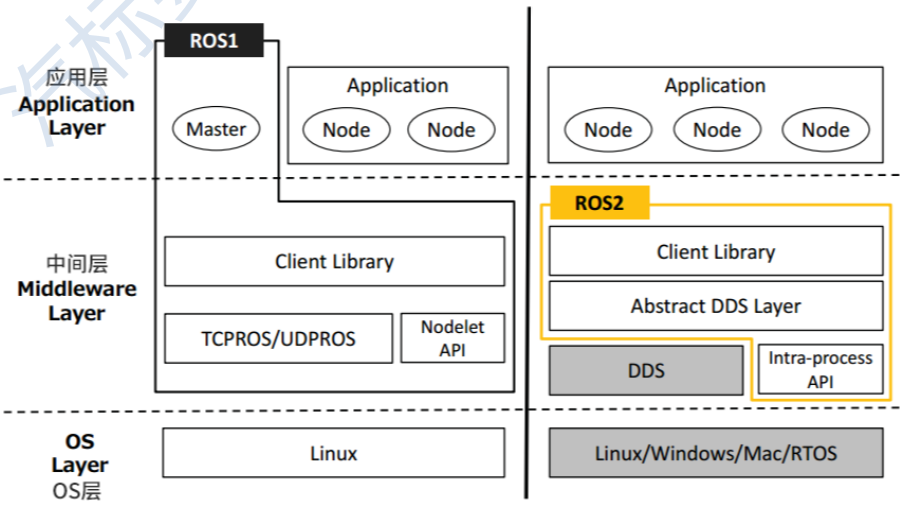

图 8 ROS 架构（来源于网络）

目前普遍采用的车控操作系统底层内核主要有 Linux、QNX 和其他 RTOS（如 FreeRTOS、ThreadX、VxWorks 等），三者之间的主要特点对比如表 2 所示。

Linux 最初是作为通用操作系统而设计开发的，但提供了一些实时处理支持，这包括大部分 POSIX 标准中的实时功能，支持多任务、多线程，具有丰富的通信机制等。除此之外，Linux 社区有实时性增强 patch，在 Linux 内核原有 RT 功能上，增加了中断线程化、优先级默认继承等功能。Linux 也提供了符合 POSIX 标准的调度策略，包括 FIFO 调度策略、时间片轮转调度策略和静态优先级抢占式调度策略。另外，Linux 还提供了内存锁定功能，以避免在实时处理中存储页面被换出，同时提供了符合 POSIX 标准的实时信号机制。

QNX 是一种商用的遵从 POSIX 规范的类 Unix 实时操作系统，其主要特点是符合分布式、嵌入式、可规模扩展的硬实时操作系统。QNX 遵循 POSIX.1 (程序接口）和 POSIX.2 (Shell 和工具)、部分遵循 POSIX.1b（实时扩展)。QNX 的微内核结构是它区别于其它操作系统的显著特点。QNX 的微内核结构，内核独立自处于一个被保护的地址空间；驱动程序、网络协议和应用程序处于程序空间中。

表 2 车用操作系统内核比较

项目指标	Linux	QNX	其他 RTOS
实时性能	需要进行实时性改造	微秒级延时	微秒级延时
开放性	源代码开放	封闭	商用或开放
许可协议	GPL	商用	N/A
费用	无授权费用（商用收费）	Royalty&License	较低或免费
功能安全	ASIL B 有可能	ASIL D	N/A

软件生态	应用生态链完善	汽车领域应用广泛	有限
优势	技术中立，支撑复杂功能	性能强，安全性高	实时性好，启动快
劣势	系统复杂	进程间通信、系统调用开销等	进程间通信、系统调用开销
主要适用范围	智能座舱、信息娱乐、TBOX、ADAS、某些域控制器等	仪表盘、智能座舱、信息娱乐、导航、ADAS、域控制器等	仪表盘、ADAS、整车控制器等

随着自动驾驶技术的快速发展，汽车对软件特别是操作系统的变革需求越来越高，主机厂、Tier1 供应商和自动驾驶软硬件技术方案提供商纷纷投入大量的人力、物力和财力进行车控操作系统的研发，希望在软件定义汽车的时代能够占据一席之地。下面将对目前国内外主流车控操作系统的开发和应用情况进行简单介绍。

（1）特斯拉 Autopilot 自动驾驶软件架构

众所周知，特斯拉是自动驾驶技术和产业化的领跑者，其优势在于以计算平台为核心，自研并领先芯片硬件、操作系统平台软件等。特斯拉自动驾驶软件架构如图 11 所示，主要特点是其操作系统基于单一 Linux 内核，打造了整套自动驾驶的软件方案，分别完成了从感知、决策规划和控制系统解决方案。从现在公开的信息可知，系统基于 Ubuntu 进行裁剪，对 Linux 内核进行了实时性改造，这个内核也开源在 github 上，深度学习框架基于 PyTorch，实时数据处理基于开源流处理平台 Kafka，拥有 48 个独立的神经网络进行多维度数据处理，并且具备强大的 OTA 升级能力。其 FSD（Full Self-Driving）计算平台硬件集成了智能座舱域和自动驾驶域，操作系统通过 OTA 软件升级，充分利用数据、云计算生态，开创汽车产品价值和服务的新

模式。

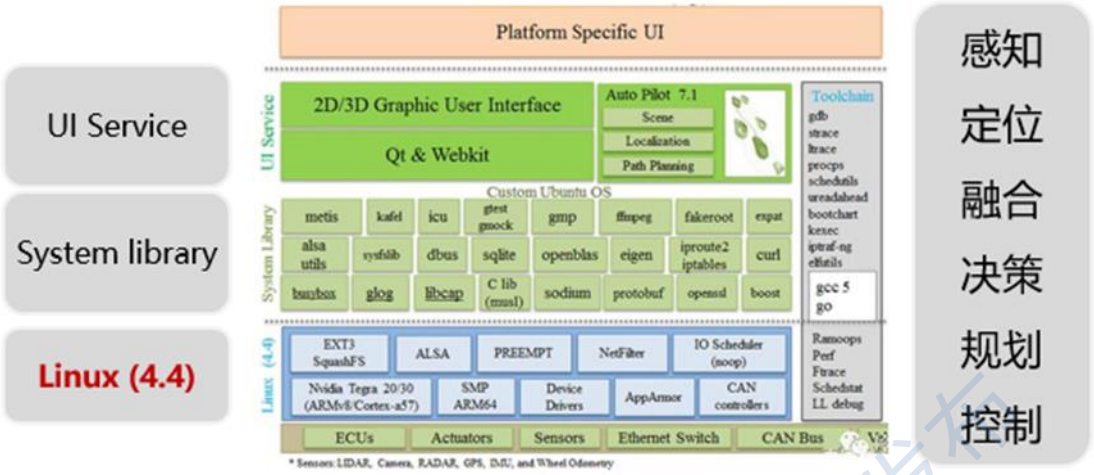

图 9 特斯拉 Autopilot 软件架构（来源于网络）

Autopilot 与娱乐控制层掌控了所有的摄像头和雷达传感器。在模块内部，Autopilot 系统和娱乐系统这两大部分通过 CAN 和高速串行总线 FPD-Link 打通，两者之间甚至可以传递视频数据。Autopilot 数据流处理机制如图 10 所示，融合自动驾驶与车内感知，实现服务驾驶闭环。

图 10 Tesla Autopilot 数据流处理机制

通过 E/E 架构的集中化，特斯拉将汽车的软件开发内化，将汽车底层硬件标准化和抽象化，此举让特斯拉通过软件定义汽车和创新变得更容易。特斯拉的 Autopilot 进化沿着功能集中化、资源共享化的道路前进，体现了特斯拉软硬件解耦，通过软件定义汽车的实践。

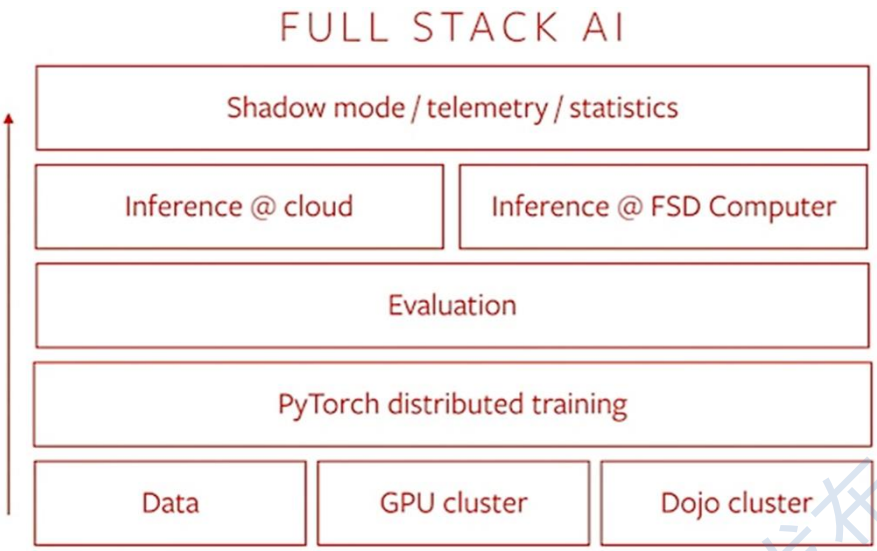

图 11 特斯拉全栈 AI 架构（来源于网络）

特斯拉在打造 Autopilot 整体软件栈时采用的理念（参见图 11）：站在巨人肩膀上进行创新，充分利用开源项目进行全栈开发，从开发（比如 UI 框架基于 QT，前端也基于一些开源的库）、构建、到部署都采用了开源的方案，推动车端算力服务、调度机制，以及云端算力资源优化布局，培育智能驾驶新业务模式。

最底层的是数据、GPU 集群以及 Dojo 计算集群，这一层主要进行数据采集、标注和训练，生成算法模型；采用基于 PyTorch 开源框架的深度神经网络对模型进行分布式训练；用损失函数对模型进行评估；在评估层之上，是云端推理和车端 FSD 芯片推理，到这一层，意味着算法模型走完了大部分流程，然后就是部署到车端；在车端，这是车控操作系统运行范围，特斯拉通过影子模式将这些算法模型与人类驾驶行为进行比对，检测是否存在异常。

这样从数据采集到算法部署的闭环，随着更多汽车上路展开数据收集，可实现基于海量数据的驱动，让系统性能不断迭代，更加优秀。

在这个闭环当中，涉及到数据集、模型训练神经网络、云端和车端推理算法等要素。

总结来看，软件定义汽车深刻地改变了汽车行业的盈利模式，将高性能的硬件预埋作为投资，通过软件更新服务盈利，已经成为特斯拉为代表的新造车势力的标准操作。

（2）大众中央集中式软件参考架构

大众汽车为了加速自动驾驶技术的应用，组建了庞大队伍自主开发汽车操作系统 vw.OS。 vw.OS 采用的是基于 Adaptive AUTOSAR 面向服务的软件架构，其中，中央集中式软件参考架构如图 12 所示。大众新一代 EE 架构的设计特点主要有：1）采用高性能处理器、高速网络；2）兼容 POSIX 的内核（Linux/QNX 等）Linux + Adaptive AUTOSAR 操作系统；3）应用软件和 I/O 功能解耦，减少整个系统的复杂性和应用之间的依赖性；4）高效、快速地开发用户功能；5）采用面向服务的通信。

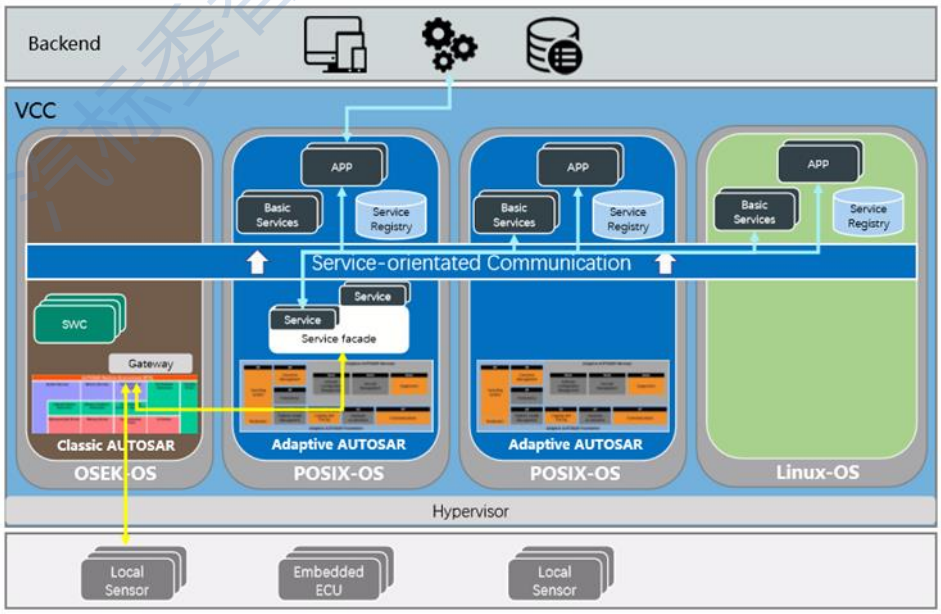

图 12 大众中央集中式软件参考架构（来源于网络）

（3）华为 MDC 智能驾驶计算平台架构

华为 MDC（Mobile Data Center：移动数据中心）定位为智能驾驶的计算平台。此平台集成华为在 ICT 领域 30 多年的研发与生产制造经验，基于 CPU 与 AI 处理器芯片，搭载智能驾驶 OS，兼容 AUTOSAR，支持 L2~L5 平滑演进，结合配套的完善工具链，客户或生态合作伙伴可灵活快速的开发出针对不同应用场景的智能驾驶应用。华为 MDC 智能驾驶计算平台（以下简称华为 MDC 平台），性能强劲、安全可靠，是实现智能驾驶全景感知、地图&传感器融合定位、决策、规划、控制等功能的汽车“大脑”。适用于乘用车（如拥堵跟车、高速巡航、自动代客泊车、RoboTaxi）、商用车（如港口货运、干线物流）与作业车（如矿卡、清洁车、无人配送）等多种应用场景。华为的 MDC 智能驾驶计算平台架构主要特点有：1）提供软硬件解决方案，且高度解耦，可独立升级，硬件升级路线和软件升级路线分别独立；2）对主流传感器的适配性好，支持主流 GNSS、IMU、摄像头、激光雷达和毫米波雷达等传感器的数据接入，且支持摄像头和激光雷达点云的前融合；3）对主流中间层软件的适配性很好，可兼容 ROS 和 AUTOSAR，支持 Caffe 和 TensorFlow 等常用深度学习框架；4）核心组件（芯片、操作系统内核）自主可控；5）华为是业界唯一同时拥有 CPU 与 AI 芯片研发能力的厂家，MDC 平台硬件集成具有 CPU 与 AI 计算能力的强大 SoC 芯片，为智能驾驶提供可扩展的异构算力；6）功能软件基于 SOA 架构，遵循 AUTOSAR 规范，定义了智能驾驶基本算法组件（如感知算法组件、融合算法组件、定位算法组

件、决策算法组件、规划算法组件、控制算法组件等）的调用框架与组件之间的软件接口；上层场景应用可以灵活选择不同的算法组件组合，实现具体的场景应用功能；7）提供安全可信，高效便捷，灵活开放的应用开发端到端工具集，支持可视化&拖拽式操作及自动代码生成，可一站式开发、测试、调优，帮助客户或生态合作伙伴快速开发满足 AUTOSAR 规范的智能驾驶应用。

（4）英伟达自动驾驶平台架构

英伟达（NVIDIA）是全球领先的人工智能计算公司，利用其先进的硬件芯片开发优势，以行业较领先的高性能安全芯片为核心，提供完整的硬件平台和基础软件平台，其架构如图 13 所示。NVIDIA 计算平台硬件目前处在“Xavier”阶段，下一代平台“Orin”已发布但并未上市。Xavier 是 NVIDIA 首次生产的车规级系统级芯片，该芯片采用了六种不同类型的处理器，包括 CPU、GPU、深度学习加速器(DLA)、可编程视觉加速器 (PVA)、图像信号处理器(ISP) 和立体/光流加速器。基于 Xavier 芯片，NVIDIA 提供面向自动驾驶开发的 DRIVE AGX Xavier™，算力达到 30 TOPS，面向 L2+和 L3 级自动驾驶；提供 DRIVE AGX Pegasus™ 使用两块 Xavier 系统级芯片和两块 Turing GPU，算力达到 320 TOPS，面向 L4 级和 L5 级自动驾驶。NVIDIA Drive 的主要特点有：1）软硬件解决方案，且高度解耦，可独立升级，硬件升级路线和软件升级路线分别独立；2）硬件优势明显，是 GPU 设计、生产领域的领导者；3）软件生态非常好，有业界最完善的官方开发套件，开发者社区相对完善；4）软件层面开放程

度较高，可在 DriveWorks（功能软件层）开放 API，也可在 Drive AV 和 Drive IX（应用软件层）开放 API；5）系统软件层融合了第三方 RTOS+AUTOSAR，设有 Hypervisor 层，第三方量产 RTOS 方案通过 ASIL D 认证；6）算法加速全部基于自身 CUDA 架构和 TensorRT 加速包，二者是 NVIDIA 独有，因此其软件开发生态不可脱离其硬件平台。

图 13 英伟达自动驾驶平台架构

（5）百度 Apollo 开放平台架构

百度 Apollo 是一套软件平台，其依赖的计算平台硬件需要采用第三方的 IPC，Apollo 开放平台架构如图 14 所示。百度自行研发了两款辅助性硬件 ASU（Apollo 传感器单元）和 AXU（Apollo 扩展单元），其中，ASU 用于收集各传感器的数据，通过 PCIe 传输至 IPC，此外，IPC 对车辆的控制指令也需通过 ASU 向 CAN 发送；AXU 用于满足额外算力、存储的需求，以 GPU、FPGA 形式接入已有硬件平台。百度 Apollo 的主要特点有：1）为网联云控（V2X）进行软硬件端到端的开发；2）提出“认证平台”的概念，包括车辆认证、硬件认

证；3）很好地融入了云服务，其中包括众多百度自家的其他产品，如：基础百度云服务、在线仿真产品、高精度地图、小度助手（Duer OS），各产品间彼此受益；4）由于开源，核心的算法模块在 Github 进行长时间优化后已充分产品化；5）主要侧重系统软件的开发，包含定制优化的操作系统、系统中间件及算法功能模块，大部分硬件则采用第三方方案；6）产品没有涉及到 AUTOSAR 架构的额外开发适配，也无需对车辆现有的 ECU/MCU 进行改变。

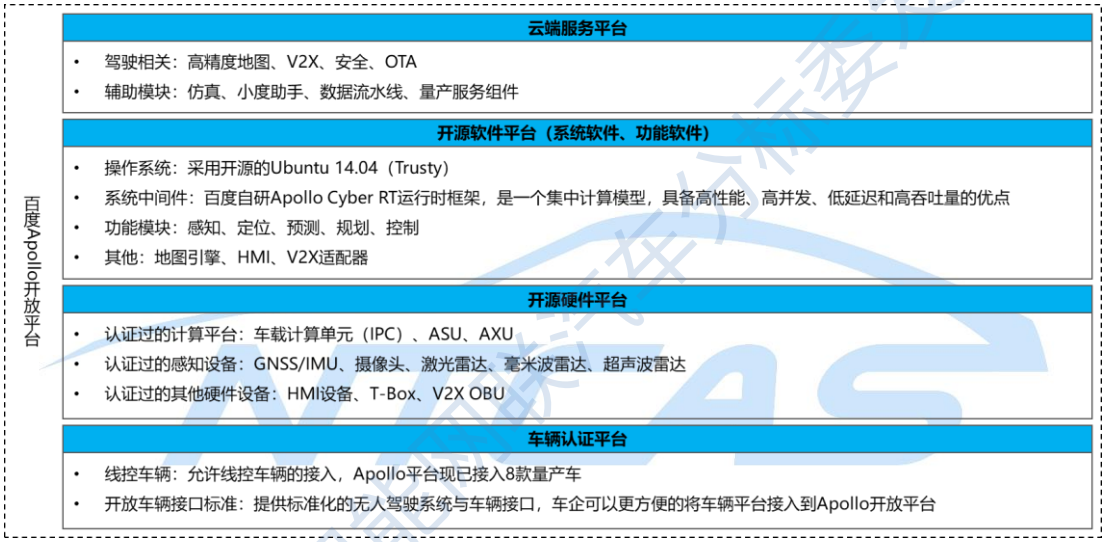

图 14 百度 Apollo 开放平台架构

2.3 车控操作系统总体技术要求研究目的及范围

2.3.1 车控操作系统总体技术要求研究目的

（1）梳理车控操作系统的国内外发展现状，明确汽车智能化、网联化发展趋势下的车控操作系统总体技术要求；

（2）研究符合“中国方案”智能网联汽车发展的车控操作系统总体技术要求，输出车控操作系统总体技术要求标准化建议。

2.3.2 车控操作系统总体技术要求研究范围

基于车控操作系统架构的研究，车控操作系统总体技术要求的研

究范围主要涉及系统软件要求、功能软件要求、面向硬件的接口要求、应用程序接口要求、系统安全要求、工具与配置类要求。通过对车控操作系统总体技术要求进行宏观定性的研究分析，为车控操作系统的产品研发提供有益参考和指导。

3 系统软件要求

系统软件是车控操作系统中支撑自动驾驶功能实现的复杂大规模嵌入式系统运行环境。车控操作系统系统软件架构如图 15 所示。

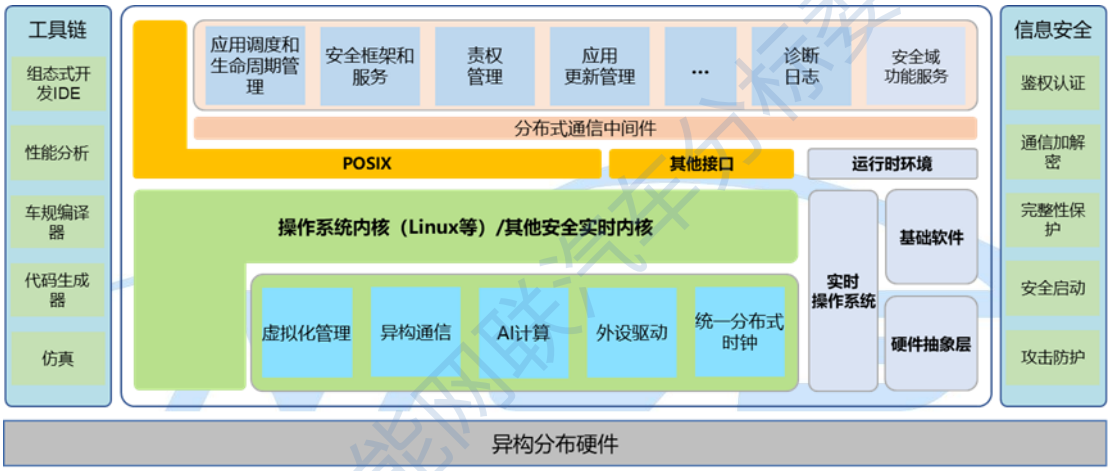

图 15 系统软件架构

车控操作系统系统软件包含智能驾驶操作系统系统软件和安全车控操作系统系统软件，以及配套的工具链和相关的网络安全措施。完善的车控操作系统方案既要求符合整车计算平台的演化，支持高性能硬件预埋，又要能支持应用功能、差异化产品开发，通过软件更新服务盈利。

面向安全车控操作系统的系统软件车规级安全实时操作系统内核，支持 MCU 等控制芯片，兼容国际主流的系统软件中间件如 Classic AUTOSAR 标准等，满足车辆动力电子、底盘电子、车身电子等实时

控制功能安全应用需求。

面向智能驾驶操作系统的系统软件以车规级操作系统内核，支持高算力计算异构芯片，以标准的 POSIX 接口为基础，兼容国际主流的系统软件中间件如 Adaptive AUTOSAR 等，满足智能驾驶不同应用所需的功能安全和信息安全等要求。

与安全车控操作系统相比，智能驾驶操作系统操作系统平台要求主要体现在如下方面：

（1）强大的计算能力，以满足图像识别和决策计算的要求；

（2）强大的数据吞吐能力，以满足多传感器数据的实时接入和处理；

（3）高度的灵活性、扩展性、可编程性，以满足多种算法模型的需要；

（4）需要快速学习和易用性，以满足 ADAS 和自动驾驶算法所需调试、调优、调测。

3.1 操作系统内核

操作系统内核需要提供操作系统最基本的功能，负责管理系统的进程、内存、设备驱动程序、文件和网络。车控操作系统内核一般需要满足以下要求：

（1）高实时性

车控操作系统要求具有高实时性的特性。这种高实时性，一方面表现在系统任务调度的时钟周期要在毫秒级，另一方面表现在高优先级的任务不能被低优先级任务所阻塞。

（2）高可靠性

车控操作系统要求能够长时间稳定运行，运行期间的系统功能和提供的服务均应保持可用。这就与允许死机、重启和部分功能失效的通用操作系统形成了鲜明的对比。业内将可靠性、可访问性和可服务性统称为 RAS（Reliability, Accessibility & Serviceability）特性。显然，车控操作系统内核要具有很高的 RAS 特性。一般说来，这些 RAS 特性应当包括：

1）高效、可靠的计算/存储/通信相关通道/组件的冗余/仲裁能力；

2）可预测错误分析能力；

3）关键进程监控能力；

4）错误恢复能力；

5）错误报告能力。

（3）高性能计算

考虑到内核的兼容性，智能驾驶车控操作系统内核需要向上层应用提供标准的 POSIX 接口（C/C++函数库等其它接口）。

3.2 虚拟化管理及硬件抽象

3.2.1 虚拟化管理

虚拟化是一个广义的术语，在计算机方面通常是指软件运行在虚拟的基础上，而不是真实的硬件。虚拟化带来的好处就是软硬件的解耦，这里的软件可以是应用程序、服务，甚至操作系统。虚拟化技术能带来很多以前解决不了的问题，例如服务的整合、负载均衡、动态迁移、系统快照、灾难恢复等。

虽然车控硬件是一个统一的平台，但不同的功能模块可能对应不同的实时性、算力、网络能力、功能安全等级，都在一个单一的操作系统下实现这些应用，会让操作系统难以实现、难以维护，更难以变更。如果采用虚拟化技术，让不同运行需求的应用跑在一个个独立的操作系统中，很多问题都可以迎刃而解。当引入虚拟化技术后，车控操作系统需要提供相应的虚拟化管理模块，即完成车控操作系统的虚拟化部署及虚拟环境的管理，包括管理工具、组件、快速部署，同时要兼顾性能和安全。

Hypervisor 需要具备高安全性、高实时性和高可靠性，实现硬件 CPU、内存、外设等资源在不同操作系统之间的隔离。由于服务之间的依赖性，任何一个部分的异常都可能会影响到系统的其他部分，Hypervisor 应定义明确的服务边界，有效地隔离故障，构建系统的容错能力和异常处理能力。多系统之间的高效通信机制也是虚拟机的关键技术之一，操作系统间可以建立虚拟以太网、共享内存和快速消息等通信机制。操作系统可以根据不同通信需求来选择合适的通信方式。

3.2.2 异构通信

由于 ADAS 和自动驾驶的发展，环境感知数据对系统总线带宽提出了更高的要求，原有基于 CAN、FlexRay 等协议的总线系统已经不再满足需求；以太网具有技术成熟、高度标准化、带宽高以及低成本等优势。随着近年来汽车电子化的快速发展，车内电子产品数量逐年增加，连接和互通的复杂性日益提高，以太网所具有的技术优势可以很好地满足汽车制造商对车内网络连接的需求。

异构通信需要解决多总线系统的通信方式统一问题，通过对多种传输介质的兼容，为互联互通层中间件提供统一的支撑协议。

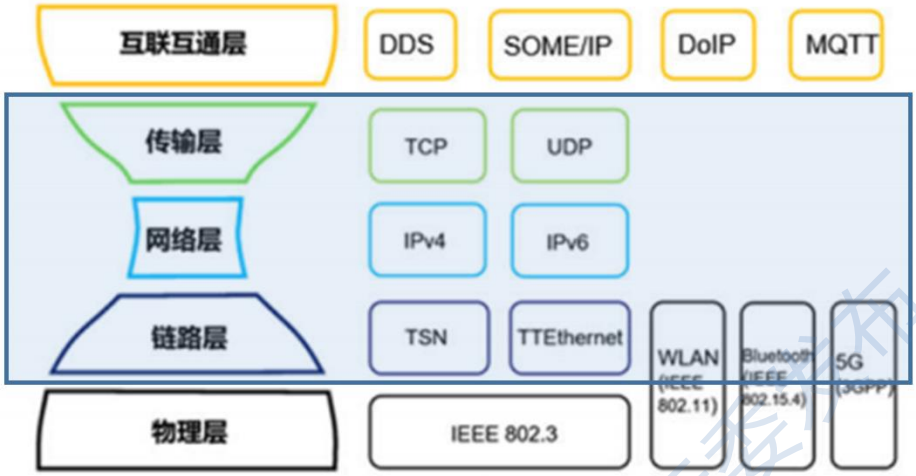

图 16 车载网络参考架构（来源于网络）

3.2.3 AI 计算

由于自动驾驶领域的高性能计算的需要，传统的 CPU 用来执行 AI 算法速度慢、性能低，用 GPU 虽然速度快，但功耗大，而且通用 GPU 不是专门针对 AI 算法开发的。因此 AI 计算领域一般使用针对 AI 算法的专用芯片，简称 AI 芯片。AI 芯片使用什么方法原理，当前在探索阶段，百花齐放，包括 GPU、FPGA、DSP 等。AI 计算模块需要广泛考虑不同种类的 AI 芯片的异同，灵活方便的对其提供支持。现有平台的 AI 芯片实现各不相同，系统软件可以提供适用于不同 AI 模式的统一 AI 算子接口，广泛考虑不同种类的 AI 芯片的适配，灵活方便的对其提供支持。

3.2.4 外设驱动

外设驱动是未被 POSIX“标准化”的特定外设的抽象。外设驱动应为这些“非标”外设定义统一的调用 API。同时为保持灵活性，外

设驱动应广泛的支持不同厂家、不同接口的外设，驱动应使用分层设计，并支持配置功能，可以根据需求灵活方便的支持不同的外设更换。

3.2.5 统一分布式时钟

系统软件需要提供整车统一的时钟，为分布式系统提供一个统一时间标度的过程，保证系统运行的实时性。车控网络中各个节点的时钟需要进行同步，保证各个节点步调一致地执行，比如控制模块和规划模块在不同节点上实现，那么这两个设备需要有相同的时钟基准，否则可能出现规划路径与控制逻辑对不上的问题。时钟同步不限于通信介质，需要支持以太网络、CAN、FlexRay 等通信协议和同步协议。

3.3 POSIX 及其他接口

POSIX 全称为可移植性操作系统接口，是一种关于信息技术的 IEEE 标准。它包括了系统应用程序接口（API）和实时扩展 C 语言。POSIX 定义了标准的基于 UNIX 操作系统的系统接口和环境来支持源代码级的可移植性。目前，POSIX 主要提供了依赖 C 语言的一系列标准服务，在将来的版本中，标准将致力于提供基于不同语言的规范。

为嵌入式实时操作系统考虑，POSIX 创建了 PSE51 子集，在 IEEE.13-2003 中进行了定义。AUTOSAR 也仅支持 PSE51，仅为应用程序提供了受限的操作系统 API。在车控操作系统中，也推荐这种受控的系统 API。

对于 POSIX 中未定义的系统功能，需要在其他接口部分定义。例如，系统定义了一系列用于系统性能监控的 API，用于应用程序自

感知其占用内存、CPU 等资源的情况。这些 API 需要在其他接口中定义，并提供完善的文档。

3.4 系统中间件及服务

3.4.1 分布式通信中间件

分布式通信中间件作为自动驾驶系统的基础框架，主要负责为上层业务模块提供协议对齐以及无主分布式通信功能。分布式通信中间件一般采用面向服务的架构（SOA）设计。

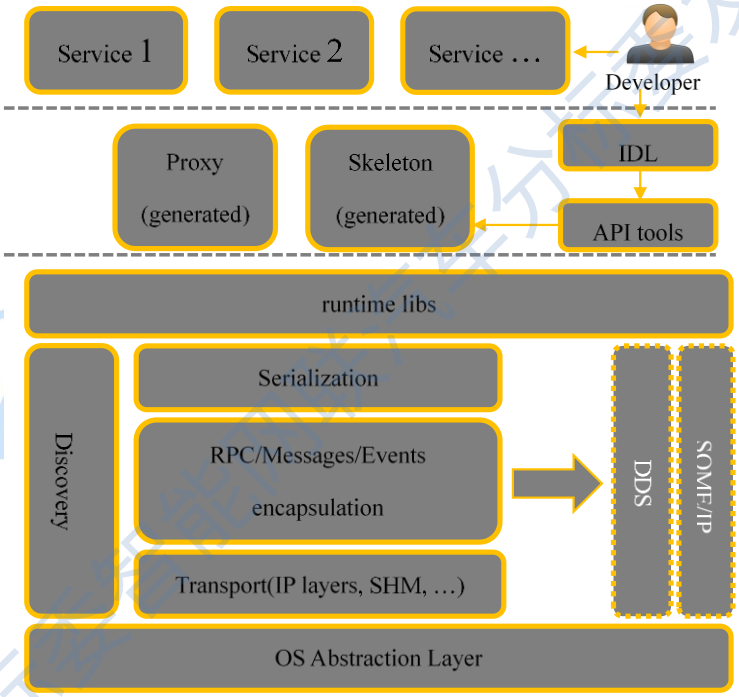

图 17 分布式通信中间件 SOA 架构

分布式通信中间件系统各层的功能要求如下：

（1）本地服务和远程服务之间的通信应以统一的接口描述语言定义的文件为契约。接口定义语言（IDL）是一种中立的接口描述语言，与具体的操作系统、编程语言无关。

（2）SOA 框架的底层核心功能应具备服务发现、消息序列化、内部事件/消息处理和传输功能。应用之间、服务之间、操作系统之间，

可通过标准的通信协议或服务接口相互通信或访问，特别是满足传感数据大数据吞吐传输。需要支持典型的车内通信协议如 SOME/IP 协议、DDS 规范等。其中，服务发现功能应具备访问控制功能，防止无权用户的窃听和侵入；传输功能应具备数据加密和签名等功能，保证通信数据的安全。

3.4.2 应用调度和生命周期管理

系统软件应支持上层应用的生命周期管理，包括运行环境初始化、应用配置加载、节点启动/重启、休眠/唤醒、正常关闭和异常处理等运行模式。

3.4.3 安全框架和服务

安全框架和服务应支持高安全、强实时性要求的系统开发。在系统软件层面需要支持基本的安全机制，包括但不限于：

（1）应用隔离保护（Freedom from Interference between Applications）；系统软件需要支持资源确定性隔离、容错、保护机制、故障隔离，支持必要的冗余备份恢复，以满足不同安全级别的应用需求。

（2）支持 Fail-Safe（指故障时系统能够导向或维持在安全状态）和 Fail-Operational（指系统失效后还可降级操作/运行）；系统软件要保护各模块加载、启动和运行的安全可靠；在异常处理方面，系统软件需要保障系统故障可识别、可监控、可隔离、可恢复。

（3）支持安全通信（E2E 通信保护）。

3.4.4 责权管理

责权管理要求操作系统实现资源标识和管理的统一 ID 体系，并支持不同层级的管理通道，包括政府安全监管级别、整车厂和供应商级别、第三方开发者级别，并内置实现并发模式下的超越管理。

3.4.5 应用更新管理

系统软件应支持持续的软件升级，实现应用模型与架构模型解耦，基础服务与硬件驱动解耦，软件组件支持根据实际要求进行动态配置与加载，满足系统软件和应用软件能够在有限的资源上完整的运行与发布。

应用更新管理应支持软件版本更新发布机制，实现应用系统可升级，包括 OS、应用、数据和模型。升级方式包括但不限于 OTA、OBD 以及其他离线升级方式。

3.4.6 诊断日志

系统软件应提供诊断机制，支持运行时、生产下线和售后诊断需求。系统软件还应支持 Logging 和 Trace 机制，提供支持开发阶段的代码调试和故障分析能力。

3.5 实时安全域

由于车辆安全环境需要和 MCU 芯片固有特性，在传统汽车 ECU 开发或未来域控制器的开发中，MCU 在相当长的一段时间内仍然是不可或缺，其主要属性为实时和安全。基于此，MCU 操作系统层面需要使用能够满足硬实时需求的 RTOS。安全车控操作系统软件支持微秒级的实时调度，和不同优先级的响应管理需求，避免资源竞争带

来的时延不确定性，确保关键应用确定性时延需求，支持和智能驾驶操作系统的互通，支持主流的通信技术满足车内、车车、车云等应用的不同通信需求。安全包含中信息安全和功能安全要求，和智能驾驶操作系统要求是统一的，也是需要满足 ISO26262、ISO21434 相关的要求。

由于 MCU 运算能力有限，因此在 MCU 平台上实现基于服务的应用开发仍然有困难，当前 MCU 应用的开发模式多基于信号实现。安全车控操作系统系统软件应具有较高可扩展性及可移植性，降低重复性工作，缩短开发周期，降低研发成本；同时，应实现模块化和层次化设计，各模块和层次应实现高内聚、低耦合的要求。

4 功能软件要求

功能软件是车控操作系统中根据面向服务的架构设计理念，通过提取智能自动驾驶核心共性需求，形成智能自动驾驶各共性服务功能模块，高效实现驾驶自动化功能开发的软件模块。车控操作系统功能软件架构如图 18 所示。

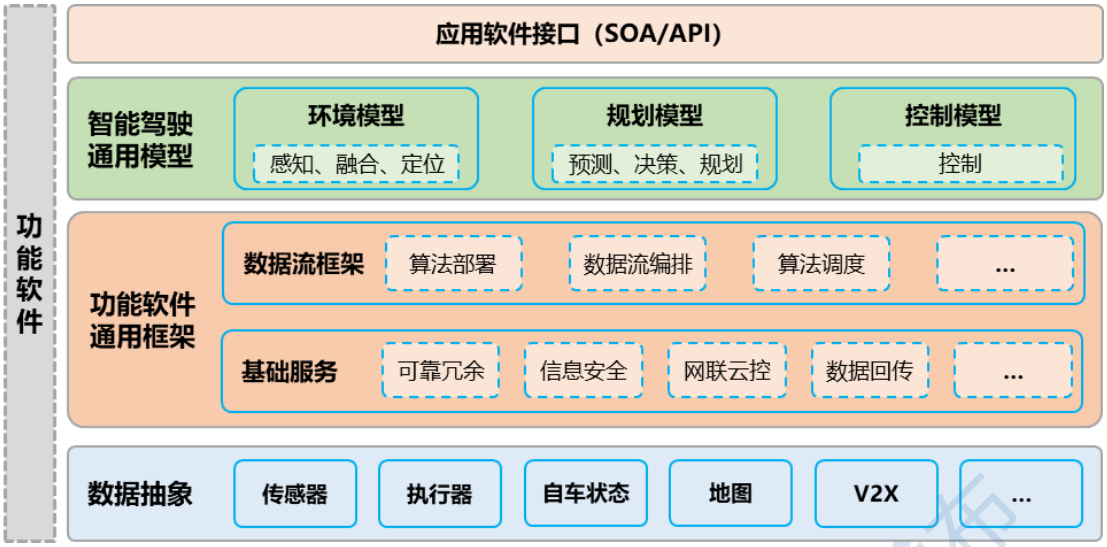

图 18 功能软件架构

4.1 智能驾驶通用模型要求

智能驾驶通用模型需要提供智能驾驶所需的感知、融合、定位、规划和控制等算法以及这些算法所需外部环境和车辆自身数据的抽象化模型。

4.1.1 环境模型要求

环境模型对驾驶环境和自车状态的语义信息（例如道路、障碍物、天气、行人、交通标志等）进行标准化描述、分类和定义和封装；提供北向接口给应用软件开发和东西向接口给规划控制算法和处理逻辑。环境模型的对外接口需要标准化并具有灵活性和可扩展性。

4.1.2 规划模型要求

规划模型需要根据环境模型信息、功能配置输入、车辆状态反馈信息预测未来一段时间内的交通参与者的运动状态，并且及时做出正确的行为决策，为运动规划提供行为策略和约束条件，最终输出符合车辆运动学和动力学约束的轨迹，同时满足实时性、安全性、舒适性的要求。

4.1.3 控制模型要求

控制模型主要承接上游规划决策及环境模型的输入，以及车辆底盘的反馈信息，计算出车辆的横纵向控制量，完成既定的动态驾驶任务，满足安全、舒适、实时、高效的驾驶要求。需要满足异构算法在不同场景下的部署要求；提供开放的输入输出接口，能够适配不同类型的上下游输入信息，满足跨车型部署及搭载的要求；提供在不同环境下的部署接口，满足跨平台搭载的要求；能够提供故障诊断、配置及标定接口，满足车规级的开发要求。

4.2 功能软件通用框架要求

功能软件通用框架是支撑智能驾驶所必需的数据处理、算法逻辑处理和运行的主干。功能软件通用框架需要做到高性能计算、高可靠性和稳定性。

4.2.1 数据流框架

数据流框架需要对智能驾驶算法进行逻辑编排和运行驱动，同时对所需数据进行实时处理和管理。数据流框架需要有高性能计算，统一的东西向上下游模块之间的接口，使在其框架上运行的算法可以根据不同的驾驶场景进行参数和算法优化或可插扩。

4.2.2 基础服务

基础服务包括智能驾驶平台上必备和通用的功能和服务。例如可靠冗余、信息安全、网联云控、数据回传等。基础服务需要满足可配置、可调试、可升级的基本要求，同时，基础服务需要做到高效、轻量级，对于智能驾驶的实时算法逻辑运行和实时数据处理做到最小影

响或干扰。

4.2.2.1 可靠冗余

可靠冗余组件需要完成对整个系统的监测和故障处理。系统中所有硬件模块和除可靠冗余组件外的其它所有软件模块，都可以作为被管理实体，接受可靠冗余模块的监测和故障处理。

可靠冗余组件分成控制平面和数据平面两个域：控制平面提供一套通讯协议和管理机制，实现可靠冗余模块和被管理实体之间的状态同步、故障监控等功能；数据平面通过与控制平面之间的状态通讯，实现数据备份、主备切换和功能恢复等功能。

4.2.2.2 信息安全

作为车端信息安全管理体系中的一个组成部分，基础服务平台中的信息安全组件需要为数据流框架上的数据流编排和算法部署等组件提供数据安全的支持。信息安全管理体系需要对车端数据的数据类型、数据安全等级和各功能算法模块的功能类型提出明确定义。信息安全管理体系中的策略管理模块在以上定义的基础上，根据用户选择、主机厂预置和监管要求，制定数据管理策略，并将相应策略下发到基础服务层的信息安全组件。当接收到算法注册或启动请求时，算法部署模块查询信息安全组件，检查算法的输入和输出数据类型、安全等级是否与算法功能类型定义匹配。算法部署组件需按安全策略检查结果执行算法部署，或终止部署并返回部署失败原因。同理，算法启动模块需要查询信息安全组件，以确保启动的数据流是被策略允许的。

4.2.2.3 网联云控

网联云控服务需要满足网联汽车准入、联网运营监管等相关标准；提供可插扩的算法和应用接口，可适配相关标准和区域性行驶场景的应用协议和迭代升级；能够支持不同的通讯能力，提供低延迟、高带宽、高可靠、高安全的车内外通信能力；能够适配不同厂商的通信硬件设备，满足跨车型部署和搭载的要求；能够提供故障诊断、功能配置，满足车规级的开发要求。

4.3 数据抽象

数据抽象通过对传感器、执行器、自车状态、地图以及来自云端的接口等数据进行标准化处理，为上层的智能驾驶通用模型提供各种不同的数据源建立异构硬件和驾驶环境的数据抽象，达到功能和应用开发与底层硬件的解耦和依赖。

数据抽象层需要满足对来自不同类型的数据源数据的统一抽象描述、标准化的数据结构定义，提供上层功能和应用开发的统一数据接口。

5 面向硬件接口要求

车控操作系统要支持智能驾驶软件的持续升级，相应的硬件也需要满足整车生命周期的软件升级和部件更换（如感知部件的升级和更换）。车控操作系统需要支持通用 CPU、GPU 等大算力计算处理器和 AI 等协处理器，支持通用的存储接口（Flash、SSD）、以太网、串口、USB、SPI、CAN 总线、I2C、I2S、PCI 等接口，支持未来新技术和接口的扩展。

5.1 CAN 总线

CAN（Controller Area Network Bus）是传统 ECU 之间实现互联互通的、面向广播的串行总线系统，满足 ISO11898 CAN 标准。标准 CAN 包含的非信息数据大于 50%，即 CAN 线上只有一半以下的数据是真正有用的信息，其他都是用于协议控制的非数据信息，因此 CAN 总线负载通常比较高，因此 CANFD 在 CAN 的基础上进行了升级。

CAN 数据传输速率为 1 Mbits/s 时，传输最大距离为 40m，速率为 5 Kbits/s 时传输最大距离为 10km，基本站点数 64，8 字节传输，传输媒体是屏蔽双绞线或光纤。CANFD 数据传输速率为 1 Mbits/s，64 字节传输。

CAN/CANFD 数据链路层协议采用对等式（peer to peer）通信方式，即使主机出现故障，系统其余部分仍可运行（当然性能受一定影响）。当一个站点状态改变时，它可广播发送信息到所有站点。

5.2 车载以太网总线

车载以太网是未来 ECU 发展趋势，ECU 之间组网方式更加灵活，支持：一对一、一对多、多对一、多对多等通信方式。与车辆外部设备相连兼容 OBD 标准。

车载以太网需支持标准以太网（10Mbits/s）和快速以太网（100Mbits/s）标准，未来将会支持千兆以太网（1Gbits/s）标准。

车载以太网使用单对非屏蔽电缆以及更小型紧凑的连接器，使用非屏蔽双绞线时可支持 15m 的传输距离（对于屏蔽双绞线可支持 40m），这种优化处理使车载以太网可满足车载 EMC 要求。

车载以太网需满足 IEEE802.1/IEEE802.3 局域网标准，音频视频的传输、定时同步等规范还需支持 IEEE1722 和 IEEE1588 等标准。面向服务架构的车载以太网协议需支持 SOME/IP、DoIP、AVB 等协议。

5.3 USB 总线

USB（Universal Serial Bus，通用串行总线）是一个外部总线标准，用于规范 ECU 与外部设备的连接和通讯，是车载设备中广泛应用的接口技术。

车载设备使用 USB 通常可以外接存储设备等，也可以实现 ECU 之间互联互通，如中控使用 USB 与 T 盒互联实现 USB-ECM、NCM profile，T-Box 可以共享无线移动互联网给中控。

USB 总线需覆盖 2.0 及以上标准，可以支持高达 480Mbits/s 传输速率。

5.4 UART 接口

UART 通常是 ECU 内部 MCU 与 MPU 实现交互信息的硬件承载接口，满足 RS232C 规格，属于异步通信方式。UART 满足全双工、高可靠数据传输与交换。

通常 UART 传输数据速度从 9600bits/s 到 4Mbits/s 可设置：低于 2Mbits/s 传输数据速率使用 Rx、Tx、GND 三根信号线就可以，2Mbits/s 以上的传输数据速率需要增加 CTS/RTS，DTR/DSR 信号线。

UART 所加载的通信协议可以在软件设计时自行定义。

5.5 LIN 总线

基于通用 UART 接口的 LIN 总线，极少的信号线即可实现国际标准 ISO 9141 规定；传输速率最高可达 20Kbits/s；单主控器/多从设备模式无需仲裁机制；保证信号传输的延迟时间；不需要改变 LIN 从节点的硬件和软件就可以在网络上增加节点；通常一个 LIN 网络上节点数目小于 12 个，共有 64 个标志符。

5.6 SPI 接口

SPI（Serial Peripheral Interface，串行外设接口）是一种同步外设接口，采用主从模式（Master-Slave），支持一个或多个 Slave 设备。

SPI 需要有以下信号线：

SDO–主设备数据输出，从设备数据输入；

SDI–主设备数据输入，从设备数据输出；

SCLK–时钟信号，由主设备产生；

CS–从设备使能信号，由主设备控制。

为了实现高速传输，通常还需要 sync 和 require 信号线。SPI 的最大传输速率由 CPU 处理 SPI 数据的能力所决定，现在已知的可以到 40Mbits/s。

5.7 I2C 接口

I2C 为半双工串行通信方式，它只需要两根线即可在连接于总线上的器件之间传送信息。

I2C 支持 100Kbits/s，快速模式下可达 400Kbits/s，高速模式下可达 3.4Mbits/s。

5.8 I2S 接口

I2S 总线接口可作为一个编码解码接口与外部 8/16 位的立体声音频解码电路（CODEC IC）相连，该总线专责于音频设备之间的数据传输。I2S 有 3 个主要信号：

串行时钟 SCLK，也叫位时钟（BCLK），即对应数字音频的每一位数据，SCLK 都有 1 个脉冲。SCLK 的频率=2×采样频率×采样位数。

帧时钟 LRCK(也称 WS)，用于切换左右声道的数据。LRCK 为“1”表示正在传输的是右声道的数据，为“0”则表示正在传输的是左声道的数据。LRCK 的频率等于采样频率。

串行数据 SDATA，就是用二进制补码表示的音频数据。

5.9 PCI-e 接口

PCI-e 是一种高速串行计算机扩展总线标准。该接口标准广泛地应用在高速闪存卡与 CPU 之间进行数据交换。支持高达 2.5Gbits/s 的数据传输速率。

5.10 RGMII 接口

RGMII 一般用于 MAC 和 PHY 之间的通信，均采用 4 位数据接口，工作时钟 125MHz，并且在上升沿和下降沿同时传输数据，因此传输速率可达 1000Mbps。

RGMII 同时兼容 MII 所规定的 10/100 Mbps 工作方式，支持传输速率：10Mb/s、100Mb/s、1000Mb/s ，其对应 clk 信号分别为：2.5MHz、25MHz、125MHz。RGMII 数据结构符合 IEEE 以太网标准，接口定义见 IEEE 802.3-2000。

6 应用软件接口要求

应用软件接口提供统一的开发环境和工具，应用软件接口对底层具体功能实现进行封装和抽象化，应用开发者可直接利用软件接口做提供的服务订阅和接口函数 API，以及所提供的各种算法库进行高效灵活的定制化开发。

应用软件接口需满足模块化、灵活性和可扩展性。针对不同的模块，例如环境模型、功能配置、算法模型等提供不同类型和颗粒度的应用接口；应用接口本身应该具有灵活性，例如可以接受不同类型的参数，运行模糊逻辑等；应用接口也应该具有可扩展性，用户可以基于现有的接口进行增强和优化。

7 系统安全要求

7.1 系统安全共性要求

7.1.1 安全技术要求

7.1.1.1 安全隔离与保护

根据目标系统的信息安全（等级）要求，车控操作系统可选择实现下列的安全隔离机制与保护功能：

（1）基于处理器的存储保护单元（MPU）或存储管理单元（MMU），实现基本的存储分区机制，以及不同安全级别存储空间的访问控制；

（2）支持操作系统与应用之间的隔离、操作系统之间的隔离、安全应用与非安全应用之间的隔离、以及不同功能安全等级应用之间

的隔离，防止功能安全故障的传播，控制应用对操作系统代码、数据和堆栈空间的访问，并提供应用间安全通信机制（例如，采用复制消息的方式消除直接共享内存带来的漏洞风险）；

（3）操作系统内核代码应运行在必要的处理器特权级（模式）下；

（4）针对高度集成的、多功能域的系统，提供分域隔离及域间安全通信机制；

（5）基于处理器的虚拟化扩展（例如 ARM 处理器的 Hypervisor 模式），操作系统实现虚拟化机制，针对存储分区、I/O 分区以及处理器运行时间提供全方位的虚拟机管理功能，实现不同安全等级虚拟机之间的安全隔离，以及虚拟机切换时的安全处理（例如清除缓存信息、预加载信息等）。

7.1.1.2 实时调度与安全切换、响应确定性

（1）提供基于优先级的实时调度机制，尤其需要考虑避免恶意进程/线程绕过内核调度机制，长时间持续占用 CPU 资源，导致其他用户/主体无法获取 CPU 资源。提供防止优先级反转的机制，防止因资源竞争可能导致的死锁；

（2）操作系统服务的运行时间具有确定性，即系统服务的运行时间与系统负载（例如活动进程、线程的数量）无关；

（3）安全切换（残留信息清除）：恶意软件可能利用操作系统对于残留信息的处理缺陷，在执行过程中对未删除的残留信息进行利用，以获取敏感信息。应保证重要的数据信息在进程、线程、任务切换后

被安全处理，不会留下可被攻击者利用的残留数据信息，防止敏感信息泄露。

7.1.1.3 系统服务保护

应用软件调用操作系统服务时，对其调用的上下文、调用参数等进行检查，以防止对系统服务不正确或非法的使用。

7.1.1.4 软件安全升级与更新

车控操作系统应支持本地或远程的软件安全升级过程，包括操作系统自身的升级，对升级软件包提供授权检测、完整性保护、机密性保护等功能，并可通过软件更新修复功能安全方面的故障与缺陷。具体要求：

（1）软件提供方使用数字签名、哈希及加密技术对软件升级包进行授权保护、完整性保护和机密性保护，车辆端的安全升级功能对升级包进行相应的授权检测、完整性检测和解密；

（2）在升级失败时，系统应能够回滚，并保证其完整性。

7.1.2 开发和应用过程安全要求

车控操作系统的设计开发过程应考虑根据其具体的目标应用系统，根据功能安全危害分析、信息安全威胁分析和风险评估的结果，分别确定车控操作系统在功能安全和信息安全方面需要提供的安全机制与功能要求，以及在操作系统自身的设计、实现、测试、验证等开发阶段的活动要求以及所采用的方法与措施，以满足有关标准、法规（例如功能安全相关的 ISO 26262-2018、SOTIF 和 RSS，信息安全相关的 ISO/SAE 21434 等）对于汽车软件安全生命周期过程的要求。

具体包括但不限于：

（1）对于操作系统这样的通用软件而言，需要根据 ISO 26262-2018 中有关 SEooC 所定义的开发流程要求，对车控操作系统的典型应用场景做出假设，并基于此开展相关的安全风险分析和评估，识别安全目标、需求等内容。未来在将此操作系统与具体安全相关系统/应用相结合的时候，应保证假设的条件与实际情况的一致性。根据对安全等级的分解，车控操作系统中的软件部件需要达到应用所要求的 ASIL 等级（如高级辅助驾驶决策的应用要求 ASIL-D 的安全等级，则车控操作系统中对应的软件部件也需要达到不低于 ASIL-D 的安全等级）；

（2）对于车控操作系统在信息安全方面的目标与需求分解，也应参照类似方法，对其目标应用系统进行分析；

（3）车控操作系统的代码实现遵循安全相关的编码规范（例如 MISRA、CERT C Secure Coding Standard 等）的要求；

（4）对操作系统代码实施安全性检测（基于适宜的安全规则），并开展动态安全测试（信息安全方面可包括渗透测试）；

（5）操作系统应用的安全要求

结合车辆设备系统在功能安全等级及信息安全方面的具体要求，以及设备自身软硬件资源情况、性能需求等考虑对车控操作系统应用的安全要求，具体如下：

1）操作系统从自身层面提供的安全能力是有限的，系统软件整体的安全性还有赖于应用对操作系统机制和功能的正确理解和应用。

因此，操作系统提供方应向应用开发方明示对于操作系统的正确使用事项和要求（通过提供内容详尽的用户手册，向车控操作系统的应用开发方说明安全相关的、正确的或适宜的使用信息。）；

2）操作系统的安全机制、策略和功能应合理配置，以确保应用能够按照预期的方式和安全策略正确执行操作系统的相关操作。

（6）第三方软件集成安全

车控操作系统内会使用到众多第三方软件部件和成熟的开源软件。在应用这些软件前，需要明确这些软件的安全需求，然后对其进行安全分析并及时对安全风险采取规避或改进措施。在将这些软件部件集成到车控操作系统中后需要对其安全需求进行验证。

7.2 功能安全要求

7.2.1 总则

车控操作系统发生故障时，功能失效可能会导致车辆潜在的危害事件，所以车控操作系统为功能安全相关系统，其生命周期内应满足功能安全要求，以避免或降低因故障所导致的风险。

本章节规定了车控操作系统在功能安全方面的流程、策略、活动、文档及测试验证的特殊要求。

本章节不针对车控操作系统的标称性能，也不作为车控操作系统功能安全开发的设计指导，而是规定设计过程中应遵循的方法和要求，确保车控操作系统满足智能汽车的功能安全应用。

7.2.2 流程要求

车控操作系统宜采用 SEooC 的功能安全开发方式，开发流程应

满足 ISO26262-2018 功能安全标准 ASIL-D 的要求。

7.2.3 安全策略

车控操作系统应采用安全设计原则，在不同的应用场景下，支持“故障-安全”和“故障-可操作”的安全策略。

7.2.3.1 堆栈安全

通过提供堆栈监测等机制在一定程度上实现堆栈安全，防止出现因堆栈溢出导致的软件错误和系统故障。

7.2.3.2 时间相关安全机制

包括同步时基、应用的执行时间保护等。

（1）同步时基：为车内总线网络中的 ECU 提供同步时基，同步车内网络中各 ECU 以及 ECU 内部可执行块的执行，确保软件部件的时间确定性，检测和控制时间偏差并防止其蔓延；

（2）定时保护：对应用的执行体（进程/线程/任务、中断服务例程）的执行时间进行监测，如果偏离了预期的截止时间，则实施相应的处理以防止其影响其他执行体的执行时间。

7.2.3.3 程序流监控与看门狗机制

防止因应用软件或硬件的故障导致的系统失效，或不合理地占用 CPU 时间。

7.2.3.4 错误与失效处理机制（或故障在线监测和处理机制）

监控车控操作系统内部要素或外部相关要素中可能导致违背安全目标的故障，发现故障后应在指定时间（例如故障容错时间间隔 FTTI）内采取适宜的措施（例如重启、状态保存与恢复）使车控操作

系统进入安全状态。

7.2.3.5 冗余可靠、健康监控等机制

车载智能计算平台具备功能安全的备份冗余设计，如感知子系统、刹车/转向等系统都进行冗余设计，这就要求车控操作系统支持冗余设计，并确保冗余设计下的实时性、功能一致性等系统基本的性能。

7.2.3.6 端到端通信安全保护

在 AUTOSAR CP 规范中提出了面向功能安全的端到端通信保护功能：消息发送方负责对发送的功能安全相关消息实施保护（通过添加特定的控制数据），接收方负责对所接收消息的验证（例如采用 PDU 计数器用来检测消息的“失序”、“丢失”和“重复”），如果发现错误则调用相关的处理程序。造成消息出错的硬件原因可能是：处理器错误、网络硬件故障、电磁兼容问题、处理器核间通信错误等；软件原因可能是：系统调用接口层错误、通信软件错误、网络协议栈错误、操作系统错误等。

7.2.4 活动及文档要求

7.2.4.1 输入信息要求

车控操作系统功能安全开发前需要以下输入信息，这些信息可以来源于外部需求或者基于分析假设，包含但不限于：

（1）技术安全概念（TSC）

a）技术安全需求（TSR），详细说明系统要素和接口的功能、依赖性、约束、属性和实施的安全机制；

b）系统安全架构示意图、各个安全相关要素及内外部接口。

（2）车控操作系统功能技术规范

车控操作系统功能需求及初步的架构框图，详细说明操作系统各功能的逻辑、要素和内外部接口。

（3）软硬件接口（HSI）规范

软件和硬件的交互，说明硬件的运行模式、配置参数、资源配置、访问机制、时间限制及诊断机制。

示例 1：硬件的运行模式，如默认、初始化、测试或高级模式。

示例 2：配置参数，如增益控制、带通频率或时钟预定标器。

示例 3：资源配置，如内存映射、寄存器分配、计时器、中断、I/O 端口。

示例 4：访问机制，如串行、并行、从机、主/从机。

7.2.4.2 开发活动及文档要求

（1）车控操作系统的软件安全需求

车控操作系统的软件安全需求来源于假设的技术安全需求分解或者来源于软件功能的安全分析，软件安全需求应支持最高 ASIL-D 的安全目标。

（2）车控操作系统的软件安全架构

详细定义车控操作系统软件架构、各个安全相关软件组件、内外部接口和关联的软件安全需求。软件架构的设计和文档应遵循 ISO26262-2018 标准的相关要求。

（3）车控操作系统的软件单元设计与实现

车控操作系统的软件单元设计应基于软件架构设计的软件组件，

通过流程图等方式详细定义软件单元内执行子程序和函数的执行逻辑与顺序，同时保证软件单元之间接口的一致性。车控操作系统软件代码编写应遵循编码规范。软件单元设计与实现和文档、代码应遵循ISO26262-2018 标准的相关要求。

（4）车控操作系统的安全分析

车控操作系统通过安全分析从总体上说明对影响功能安全的故障或故障组进行了有效识别和处理，以此来支持上述活动和文档。

安全分析可采用潜在失效模式与影响分析(FMEA)、故障树分析(FTA)或适合安全分析的其他类似方法。

7.2.5 车控操作系统的验证和测试

（1）应对车控操作系统软件进行单元验证，以确认车控操作系统软件单元设计满足相关的安全需求；单元验证通过代码检查、静态分析和动态单元测试等方式实现，具体要求和方法应遵循 ISO26262-2018 标准的相关要求；

（2）应对车控操作系统软件集成进行验证，以确认集成的软件单元和组件设计满足相关的安全需求，具体要求和方法应遵循ISO26262-2018 标准的相关要求；

（3）应对车控操作系统进行嵌入式软件测试，以确认车控操作系统在目标环境中运行符合相关的安全需求，硬件在环测试、实验车环境下测试及整车测试，具体要求和方法应遵循 ISO26262-2018 标准的相关要求；

（4）车控操作系统在实际应用中需要确认实际应用场景、系统

功能和性能要求与其SEooC开发过程中所基于的输入信息的一致性，确保操作系统的安全应用。

7.3 信息安全要求

信息安全技术覆盖的安全需求包括：真实性、机密性、完整性、可用性、防抵赖、可授权。

（1）真实性：防止攻击者冒充某人或某事；

（2）机密性：是指网络信息不被泄露给非授权的用户、实体或过程，即信息只为授权用户使用。需要防止攻击者通过访问传输过程或是存储的数据而导致的信息泄露，保护隐私信息；

（3）完整性：指在传输、存储信息或数据的过程中，确保信息或数据不被未授权的篡改或在篡改后能够被迅速发现。需要防止攻击者改变传输过程或是存储的数据，或是改变通过软件、固件或硬件实现的功能；

（4）可用性：产品在规定的条件下和规定的时刻或时间区间内处于可执行规定功能状态的能力。需要防止攻击者中断系统合法操作的行为，使得系统功能是持续可用的。从车控操作系统的层面，可用性主要考虑保障内核管理的资源的可用性，不被恶意程序不合理占用资源，而影响系统安全相关功能；确保通信资源的可用性，不被恶意流量占用而影响正常的通信功能。

（5）防抵赖：防止攻击者的行为不能被追溯；

（6）可授权：防止攻击者执行未被授权的行为。

涉及车控操作系统的信息安全主要技术要求如下。

7.3.1 可信执行环境

基于处理器的安全扩展例如 ARM 处理器的 TrustZone，实现安全可信的执行环境（TEE），隔离非安全与安全的应用，为安全关键业务提供安全防护保障。

在“安全世界”下处理器运行于特定的特权模式下，系统的其他硬件资源（包括存储空间、设备端口等）也区分为两个世界，处理器通过响应监控器异常来实现两个世界之间的切换。安全相关的应用运行在安全世界中，并受到安全操作系统（或可信内核）的管理。可信执行环境在智能手机领域已受到广泛应用，例如在涉及用户隐私安全的应用中。

7.3.2 密码应用支撑

基于密码安全芯片的密码运算及其他功能，实现对安全芯片的能力封装，支持密码技术在系统安全中的应用。提供密码服务方面的软件栈，自下而上可包括密码安全芯片硬件驱动、密码硬件抽象、密码服务管理等。

提供密钥管理、证书管理等功能：

（1）证书管理：对设备出厂时预置的授权证书进行管理，提供证书安全访问接口；

（2）密钥管理：提供密钥安全访问的接口。

7.3.3 访问控制与身份鉴别

（1）提供访问控制机制，防止安全相关功能、数据及资源（包括各种软件硬件端口资源）等在未授权情况下的访问；

（2）遵循访问权限最小化原则设置安全相关资源（包括敏感信息）的访问控制权限；

（3）提供适宜的身份鉴别机制如口令鉴别、生物特征鉴别、数字证书鉴别或图形鉴别供应用使用。

7.3.4 车内总线安全通信

针对车内总线网络如 CAN、CANFD 或以太网，提供安全通信机制。

例如 AUTOSAR 标准的 SecOC 为车内网络 ECU 之间的通信提供了消息完整性、抗重放方面的能力。在协议数据单元层面（PDUs）SecOC 与 AUTOSAR 的通信系统协同，PDU Router 负责对接收的消息进行路由，对发送的安全相关协议数据单元提供给 SecOC 模块。SecOC 随后增加或是处理安全相关信息后，把协议数据单元返回给 PDU Router，由 PDU Router 进行进一步的路由处理。此外，SecOC 使用密码服务体系提供的密码服务。对于车内以太网的安全，则通过 TLS 和 IPsec 提供真实性和机密性的通信服务。

7.3.5 安全外部通信

（1）对通过 4G/5G、蓝牙、WIFI、V2X 专用接口、NFC、GPS、射频等外部访问或通信接口的通信，根据协议特定采取相应的安全策略与控制机制；

（2）身份认证：可基于 HTTPS/TLS 安全通信协议（TLS1.2 或以上，支持必要的加密套件），实施车辆端设备与远程通信对方（如云端服务平台）的双方身份验证、通信数据加密、通信数据完整性保

护等机制对通信过程实施安全防护。

7.3.6 安全可信启动

安全可信启动是基于完整性校验机制确保车辆设备启动环节的安全可信，防止系统固件、软件被篡改。通过在硬件安全芯片中预置安全启动的唯一根密钥，为系统的引导校验过程提供基础，其核心是建立可信根和逐级校验机制。主要有两种情况：

（1）MPU 系统安全启动可划分为四个阶段：可信根加载、Bootloader 引导、操作系统内核启动、操作系统服务（或其余部分）启动。系统上电后，首先可信根验证 bootloader 的完整性和真实性；接着 bootloader 验证操作系统内核的完整性和真实性，从而确保其后续的安全启动；最后验证操作系统其余部分的完整性。

（2）MCU 安全启动：可以划分成三个阶段：可信根加载、Bootloader 引导、MCU 系统程序启动。系统上电后，首先可信根验证 bootloader 的完整性和真实性；接着 bootloader 验证 MCU 系统程序的完整性和真实性。

7.3.7 系统安全

（1）系统安全防护：车辆设备开发时应采用新的操作系统版本；针对已知版本的漏洞，在开发时应完成所有漏洞补丁；车辆上市后应持续监控车控操作系统内所有软件要素的漏洞情况并及时修复发现的漏洞；禁止使用操作系统的调试版本；

（2）系统 root 防护：恶意攻击者可能通过漏洞利用等方式获取操作系统的 root 权限，进而对车辆设备系统实施木马程序植入、敏感

数据窃取等攻击。应在设备系统启动及运行时定期检测是否被 root，一旦发现系统被 root，应立即采取适宜的操作如报警提示、记录安全日志等防护措施。

7.3.8 应用安全

为车辆设备应用软件（包括第三方库）的安装、运行、卸载等实施安全管理功能。为可信的应用提供方发布的应用（简称“授权应用”）提供卸载防护、篡改防护服务，以及对非可信的应用提供方发布的应用（非授权应用）的安装进行防护。

（1）非授权应用安装防护：禁止系统安装及运行非授权应用，以防止其可能给系统带来的安全风险。应用安全模块检测到当前安装的为非授权应用时，应立即采取报警提示和终止应用安装，并记录安全日志等安全防护措施；

（2）授权应用卸载防护：禁止对授权应用的非法卸载，以防止系统不能正常提供服务的安全风险。应用安全模块检测到授权应用被非法卸载时，应立即采取报警提示、终止卸载、记录安全日志等安全防护措施；

（3）授权应用篡改防护：通过完整性检测防止运行被篡改的授权应用给系统带来的安全风险。应用安全模块检测到已安装的授权应用已被篡改时，应立即采取报警提示、终止运行等安全防护措施。

需要时，应提供用户参与应用软件安装、卸载及有关的授权功能：

1）支持用户对应用软件的安装进行授权或阻止；

2）支持用户对应用软件使用终端资源（包含通信资源和其他外

设接口）和数据进行授权；

3）在应用软件卸载时删除由其生成的数据、配置信息和用户数据。

7.3.9 数据安全

提供数据的访问控制、机密性保护、完整性保护等机制对系统及应用的敏感数据（包括用户隐私数据）进行安全防护，以防止相关数据被非授权访问、窃取、损坏而给系统及用户带来安全风险。其中机密性保护和完整性保护基于密码技术实现，机密性保护包括提供加密和解密功能，完整性保护包括哈希（散列）计算功能和完整性校验功能。

7.3.10 安全监控与防御

提供安全日志、防火墙、入侵检测和入侵防御等方面的安全功能。

（1）应对与车控操作系统安全相关的事件生成安全日志，包括对车辆设备系统的异常操作、网络访问行为尤其是对设备网络入侵和违规行为的记录。审计日志信息应被保护使其免遭篡改、非法访问及破坏；日志内容应对敏感信息进行脱敏，确保不在日志中包含任何机密数据、密钥等敏感信息；

（2）防火墙：可采用基于 IP 和端口访问控制提供黑/白名单过滤机制；

（3）入侵检测：检测 ARP 攻击、DoS 攻击、端口扫描和异常连接等异常网络行为。

7.3.11 面向业务层面安全支撑机制与功能

针对 ADAS、自动驾驶、智能座舱、蓝牙钥匙/无线钥匙、充电等汽车业务层面，车控操作系统宜从传感器安全、算法安全、数据安全等方面提供有关的安全支撑机制与功能。

例如针对自动驾驶系统，提供以下的安全支撑机制与功能：

（1）全考虑防止传感器（摄像头、激光雷达、超声波雷达、毫米波雷达、红外激光传感器、导航等）信号的干扰或篡改；

（2）考虑算法对抗的威胁；

（3）数据安全考虑用于自动驾驶计算的数据集的防篡改。

8 工具与配置类要求

面向新一代电子电气架构、智能驾驶、SOA 架构设计等发展趋势，工具链需按照标准将各个功能模块进行拆解，抽象出可配置的组件，通过后台工具完成组件、任务的配置，提高整车的开发效率，支撑软件的快速迭代。

由于智能驾驶汽车的软件越来越复杂，整个工具链需要支持车规级软件开发、测试和验证，提高整体的软件工程能力，持续集成和交付。工具链应实现以下全部或者部分功能：

- 软件全生命周期的管理，包括需求分析和跟踪；
- 模型化系统设计；
- 模型化开发；
- 模型分析和验证；

- 集成测试验证；
- 整车仿真验证；
- 代码生成工具；
- 整车级 IO 端口网络自动化配置工具；
- 调测工具，如 ISP 工具，日志诊断，性能优化工具等；
- 整车仿真工具；
- 车云工具链等。

8.1 组态式开发 IDE

组态式开发 IDE 为用户提供了一种基于可视化设计的应用程序快速开发方式（参见图 19）。

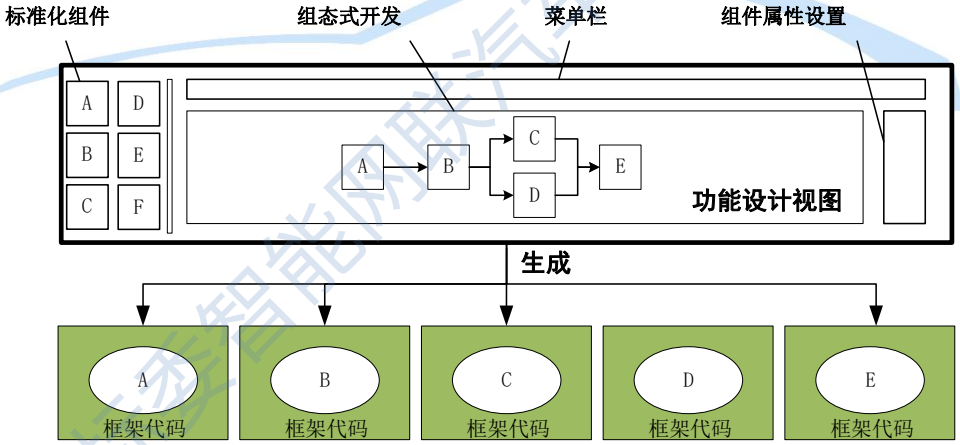

图 19 组态式开发 IDE 框架

（1）标准化组件是具有标准 I/O 接口的，与自动驾驶相关的常用功能模块；

（2）用户可以拖拽相应的标准化组件到功能设计视图中，配置其属性和连接关系，属性设置主要包括对标准化组件的入参和出参的设置，连接关系可通过绘制箭头表示；

（3）当功能设计完成以后，可通过 IDE 自动生成程序的框架，

标准组件的入参由框架代码负责获取，出参由框架代码送出。

8.2 性能分析工具

性能分析工具应具备对应用程序运行时占用的CPU、内存、网络、I/O请求等资源的统计功能，，并可以图形化的形式展现测试结果，便于用户直观地分析程序的性能，进而对程序的效率进行有针对性的优化。

8.3 车规编译器

构建自动驾驶应用程序的编译器应符合ISO-26262标准，根据智能汽车驾驶自动化等级需求，确保编译器的各种功能达到汽车完整性安全等级（ASIL）的预期级别。

8.4 代码生成器

代码生成器应能将使用中立语言描述的文件（IDL文件）转化成具体语言的类型定义代码实现协议对齐。

8.5 仿真

仿真工具应支持模拟自动驾驶算法的各种输入，并对输出进行响应和评价，同时应可支持MIL、SIL和HIL等仿真方法中的一种或多种。

9 标准化建议

车控操作系统是智能网联汽车的基础软件部分，运行于智能网联汽车车载智能计算基础平台。智能网联汽车的复杂性，需要通过车控操作系统软件架构和接口的标准化实现产业链的分工协同，提高开发效率，保障软件平台的安全可信，减少对接成本，构建统一生态。

9.1 架构要求类

架构是操作系统的顶层设计，明确了对系统实现的约束条件，为维护决策提供依据，以及便于在较高层面上实现复用。因此，将车控操作系统的架构和技术要求进行标准化有助于实现操作系统的行业共识，降低主机厂、零部件供应商等之间的沟通成本，实现操作系统软件复用，提高开发效率。车控操作系统的技术要求包括功能要求和性能要求等，通过技术要求和测试方法的标准化，可以评价车控操作系统的基本功能和性能能否满足产品设计要求。

表 3 架构和要求类标准化建议

标准化对象	分析	必要性	启动建议
车控操作系统总体架构及要求	通过标准化，可以实现域集中式计算平台智能驾驶功能的高效开发，实现可扩展、可复用。	必要	优先级高
车控操作系统性能要求及测试方法	通过规定车控操作系统的基本技术要求、功能要求和性能要求及对应的测试方法，保证产品的质量。	必要	优先级低
车用操作系统总体架构	融合了车控操作系统和车载操作系统，满足中央集中式计算平台技术发展需求。	必要	优先级低

9.2 安全要求类

功能安全和信息安全是车控操作系统产品可靠安全运行的必要组成部分。车控操作系统为智能汽车自动驾驶功能提供运行环境，其功能安全要求是保证整个系统功能安全的基础和核心。车控操作系统是单域核心攻击目标，对于车控操作系统的攻击，可以直接影响到智能网联汽车的功能安全和人身安全，也会涉及到个人隐私数据和包括测绘等敏感数据。对于车控操作系统的安全要求类的标准制定，可以

设定操作系统安全基线，最大限度保护智能网联汽车的安全运行，数据的安全存储和处理。

表 4 安全要求类标准化建议

标准化对象	分析	必要性	启动建议
车控操作系统信息安全要求	车控操作系统作为智能网联汽车的基础软件平台，信息安全要求和测试方法是保证智能网联汽车的网络安全和数据安全前提，建议尽快启动国标。	必要	优先级高
车控操作系统功能安全要求	车控操作系统作为智能网联汽车的基础软件平台，功能安全要求和测试方法是保证车辆安全运行的前提。	必要	优先级低

9.3 接口和互操作类

车控操作系统模块间接口标准化主要是为了为智能驾驶等应用提供标准化的运行环境和服务，满足不同操作系统之间以及操作系统内部不同层次之间的实现通信，高效实现和互操作，实现操作系统解耦，满足跨平台、跨车型、可扩展等要求。

表 5 接口和互操作类标准化建议

标准化对象	分析	必要性	启动建议
应用于自动驾驶功能的传感器接口	通过将感知传感器接口标准化，可以实现传感器的抽象，从而屏蔽不同类型的传感器，方便功能软件感知融合模块的算法实现。	必要	优先级低
车控操作系统面向应用程序的接口	通过向上提供标准化的面向应用程序的接口，可以屏蔽应用软件对操作系统的依赖，实现解耦，方便应用软件的开发。	必要	优先级低
车用操作系统间通信要求	通过标准化实现车控操作系统和车载操作系统的互操作性，降	必要	优先级低

	低不同开发商之间的沟通成本。		

智能网联汽车的应用正在高速发展期，车控操作系统技术作为智能网联汽车的支撑技术，具体的功能模块和接口也会随着系统的不断演进而演进，接口类标准比较倾向于市场化行为，因此可以通过制定接口类团体标准，以适应快速变化的市场需求。