车用操作系统测试评价
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 12 月

目　录

前　言

随着汽车电动化、智能化、网联化的发展，汽车操作系统已经成为车辆中最重要的组成部分之一，车辆的电子化程度决定了车辆的智能化水平、整体性能、安全性与舒适程度。随着智能网联汽车的发展，汽车在环境感知、智能决策、信息交互等方面的变革对原有的汽车操作系统提出了巨大的挑战，传统的面向实时控制的操作系统已经难以适应智能网联汽车中以图像数据和雷达数据为主的复杂信息的处理需求，采用 POSIX 接口的操作系统内核逐渐在智能网联汽车中得到广泛应用。由于智能网联汽车产业发展迅速，产品迭代速度加快，随着软件定义汽车概念的兴起，整车企业出于构建软件生态的需要，不断推出新的汽车软件平台，汽车操作系统呈现出百花齐放的发展态势。但我国车用操作系统发展仍处于初期阶段，在智能车控、车载关键核心技术研发、软硬件兼容适配、测试评价等方面仍需强化，生态支撑能力有待加强。

因此，亟需根据中国实际情况制定一套中国车用操作系统的测评标准，对相关技术进行量化测评，从而准确把控风险，同时促进成熟技术落地，推动整个汽车底层软件生态更加健康规范地发展。

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：全国汽车标准化技术委员会智能网联汽车分标委

牵头单位：中国汽车技术研究中心有限公司

参与单位：华为技术有限公司、北京地平线机器人技术研发有限

公司、惠州市德赛西威汽车电子股份有限公司、长城汽车股份有限公司、福建汉特云智能科技有限公司、上海蔚来汽车有限公司、浙江大学、北京谦川科技有限公司、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）、高通无线通信技术（中国）有限公司、襄阳达安汽车检测中心有限公司、长安汽车股份有限公司、中国汽车工程研究院股份有限公司、安徽江淮汽车集团股份有限公司、东风商用车有限公司、中国第一汽车股份有限公司、北汽福田汽车股份有限公司、上汽大众汽车有限公司、上汽通用五菱汽车股份有限公司、中兴通讯股份有限公司、江苏智能交通及智能驾驶研究院、东风汽车有限公司东风日产乘用车公司、北京百度智行科技有限公司、东软集团（大连）有限公司、江西博能上饶客车有限公司、上海机动车检测认证技术研究中心有限公司、上汽集团软件分公司、大陆投资（中国）有限公司、一汽解放汽车有限公司商用车开发院、上海智能网联汽车技术中心有限公司

参与人员：吴含冰、张路、邵学彬、鞠伟男、刘丽萍、周铮、杨继欢、杨婷婷、王胜、史松霖、马海敏、姜灿、易礼艳、周思儒、石晓坤、毛雷、陈华聪、陈文强、陈炯、张伟谦、杨国青、吕攀、孟庆洋、郭伟、郭盈、周波、李俨、陈书平、高海龙、裴健、朱乾勇、李毓强、任翔、杨茂兴、张明福、俞涛、李阳、李洋、高长胜、张晓谦、钱国平、郗冲、卜烨雯、何逸波、罗覃月、陈晓、刘翔海、殷苏辰、姜浩、贾元辉、李春林、周海龙、洪智、林宜备、费音、罗先银、刘海威、许丰、孙华、丁晓辉、余道和、韩宝广、刘时珍

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 车载操作系统 on-vehicle operating system

运行于车载芯片上，管理和控制智能网联汽车车载软件、硬件资源的软件集合，为智能网联汽车提供除驾驶自动化功能实现以外的服务，包括车载信息娱乐、网联、导航、多媒体娱乐、语音、辅助驾驶、AI 等服务。

1.1.5 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。

1.1.6 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。

1.1.7 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。

1.1.8 驾驶自动化功能 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。

1.1.9 实时安全域 real-time 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。

1.1.10 云控技术 cloud integration control technology

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”。

1.1.11 单系统架构 single system architecture

单个车载操作系统的架构，由车载操作系统内核、资源抽象层、基础库、基础服务、运行时环境、程序运行框架和车载操作系统安全模块组成，对底层硬件和上层应用程序提供统一的接口。

1.1.12 多系统架构 multisystem architecture

在同一套硬件之上运行多个车载操作系统单系统的架构，分为硬件隔离、虚拟机管理器、容器三类多系统基础架构，以及两类或三类基础架构的混合架构。

1.1.13 资源抽象层 resource abstraction layer

运行于车载操作系统内核上，为车载操作系统应用和基础服务提供 SoC 芯片平台硬件的资源抽象、整车信号的资源抽象、外部 IoT 设备的资源抽象和外围 IC 的资源抽象。

1.1.14 可信执行环境 trusted execution environment

基于硬件级隔离及安全启动机制，为确保安全敏感应用相关数据和代码的机密性、完整性、真实性和不可否认性目标构建的一种软件运行环境。其中，硬件级隔离是指基于硬件安全扩展机制，通过对计算资源的固定划分或动态共享，保证隔离资源不被富执行环境访问的一种安全机制。

1.2 缩略语

下列缩略语适用于本文件。

ACC：自适应巡航控制系统（Adaptive Cruise Control System）

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）

AR：增强现实（Augmented Reality）

ASCII：美国信息交换标准代码（American Standard Code for Information Interchange）

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

ASPICE：汽车软件过程改进及能力评定（Automotive Software Process Improvement and Capacity Determination）

AUTOSAR：汽车开放系统架构（AUTmotive Open System Architecture）

BSD：伯克利软件套件（Berkeley Software Distribution）

CA：客户端应用（Client Application）

CAF：云采用框架（Cloud Adoption Framework）

CAN：控制器局域网络（Controller Area Network）

CCA：机密计算架构（Confidential Compute Architecture）

CPU：中央处理器（Central Processing Unit）

DDS：数据分发服务（Data Distribution Service）

DMA：直接存储器访问（Direct Memory Access）

ECU：电子控制单元（Electronic Control Unit）

EL：特权级别（Exception Level）

FDBUS：高速分布式总线（Fast Distributed Bus）

FOTA：空中固件升级（Firmware Over-The-Air）

GBK：汉字编码字符集（Chinese Internal Code Specification）

HMI：人机接口（Human Machine Interface）

HUD：抬头显示（Head Up Display）

I/O：输入输出（Input/Output）

IC：集成电路（Integrated Circuit）

ICV：智能网联汽车（Intelligent Connected Vehicle）

IDL：接口定义语言（Interface Definition Language）

IoT：物联网（Internet of Things）

IPC：进程间通信（Inter-Process Communication）

IVI：车载信息娱乐系统（In-Vehicle Infotainment）

MCU：微控制单元（Microcontroller Unit）

NPU：网络处理器（Neural-network Processing Unit）

OBD：车载诊断（On-Board Diagnostics）

OEM：原始设备制造商（Original Equipment Manufacturer）

OMTP：开放移动终端平台（Open Mobile Terminal Platform）

OS：操作系统（Operating System）

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）

ROS：机器人操作系统（Robot Operating System）

RTC：实时时钟（Real_Time Clock）

RTOS：实时操作系统（Real Time Operating System）

RVC：倒车影像（Rear View Camera）

SDK：软件开发工具包外文名（Software Development Kit）

SOA：面向服务的架构（Service-Oriented Architecture）

SoC：系统级芯片（System on Chip）

SOME/IP：基于 IP 的可扩展的面向服务的中间件（Scalable service-Oriented MiddlewarE over IP）

TA：可信应用（Trusted Application）

TCP/IP：传输控制协议/互联协议（Transmission Control Protocol/Internet Protocol）

TEE：可信执行环境（Trusted Execution Environment）

TLS：安全传输层协议（Transport Layer Security）

UI：用户界面（User Interface）

V2X：车与外界的互联（Vehicle to Everything）

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

VMM：虚拟机监视器（Virtual Machine Monitor）

VR：虚拟现实（Virtual Reality）

3DES：三重数据加密算法（Triple Data Encryption Algorithm）

2 车用操作系统测试研究背景

在智能网联汽车产业大变革趋势下，汽车逐步由传统的交通工具向同时具有交通、娱乐、办公、通信等多种功能的新一代智能移动空

间和应用终端迁移。实现智能网联功能的驾驶辅助系统、车联网系统以及智能座舱系统相关电子设备逐步成为汽车电子产业研发应用的重点。从功能层面来看，汽车电子主要可以分为车身电子控制系统和车载电子装置两大类别。车身电子控制系统是利用芯片和汽车机械系统进行有机结合，对汽车的各个子系统进行控制，是保证汽车完成基本行驶功能不可或缺的控制单元，具体分为动力控制系统、底盘控制系统、车身控制系统等；车载电子装置是利用单独的电子设备，独自承担并实现其功能，对车辆的行驶性能并不会产生过多的直接影响，主要用于提升汽车舒适性和便利性，具体可分为信息系统、导航系统和娱乐系统等。

随着车身电子控制和车载电子装置功能的日益丰富以及汽车电子产品外部交互/接口标准的种类增加，这类基于微控制芯片的嵌入式电子产品逐渐需要采用类似个人电脑的软件架构以实现分层化、平台化和模块化，提高开发效率的同时降低开发成本。因此，汽车电子产品才逐步开始采用了嵌入式操作系统。操作系统是指控制和管理整个计算系统的硬件和软件资源，并合理地组织调度计算机的工作和资源，以提供给用户、其他软件接口和环境的程序集合。智能设备发展到一定程度后一般都需要专门的操作系统，例如个人电脑对应的微软 Windows 系统和苹果 Mac OS 系统，智能手机对应的谷歌 Android 系统和苹果 iOS 系统。车用操作系统是传统汽车实现智能汽车升级的关键，作为硬件核心资源管理和软件运行平台，所有应用程序都需要在操作系统上运行，围绕操作系统可以构建一个庞大的应用生态，影响

整个行业的发展。

为落实党中央、国务院关于建设制造强国的战略部署，支撑和促进智能网联汽车技术及产业发展，2019 年 10 月 25 日全国汽车标准化技术委员会（以下简称“汽标委”）在世界智能网联汽车大会正式发布《车用操作系统标准体系》。该标准规范了车用操作系统的定义、划分了车用操作系统边界、明确了车用操作系统的分类。车用操作系统是运行于车内的程序集合，其主要功能为：管理硬件资源、隐藏内部逻辑、提供软件平台、提供界面接口、为上层应用提供基础服务等。该标准将车用操作系统定义包含车载操作系统和车控操作系统，其中车控操作系统又分为安全车控操作系统和智能驾驶操作系统，如图 1 所示。车控操作系统对安全性、实时性、稳定性要求非常高，而车载操作系统更加重视开放性、兼容性。

图 1　车用操作系统分类

（1）车控操作系统

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。

目前，汽车一般采用分布式电子电气架构，即大量 ECU 单元协同工作，共同为驾驶员控制车辆提供各种功能。随着车内 ECU 数量急剧增长，汽车电子复杂程度加深，分布式架构无法满足未来汽车的应用要求，基于域控制器的新一代集成架构趋势越来越明显。随着 L3 级以上甚至无人驾驶的到来，中央处理器架构将成为发展趋势之一，多域控制器合为一体，由统一的平台来控制汽车驾驶的所有功能。智能汽车的车控操作系统主要用于车辆底盘与动力控制，以实现油门、转向、换挡、刹车等基本行驶功能，是当前重点研发的 L3 及以上级别自动驾驶功能的核心，其包含高性能复杂嵌入式系统、人工智能芯片及算法、高速网络、海量数据处理、云边协同等多种行业的融合技术。

2021 年 7 月 6 日-8 日，全国汽车标准化技术委员会智能网联汽车分技术委员会在第七届智能网联汽车技术及标准法规国际交流会（ICV2021）上正式对外发布了十项“2021 年度智能网联汽车标准化需求研究成果”，其中，《车控操作系统架构研究报告》重点介绍了车控操作系统总体架构、系统软件架构、功能软件架构以及相关标准化建议；《车控操作系统总体技术要求研究报告》重点介绍了系统软件要求、功能软件要求、面向硬件接口要求、应用软件接口要求、系统安全要求、工具与配置类要求以及相关标准化建议，车控操作系统参考架构如图 2 所示。

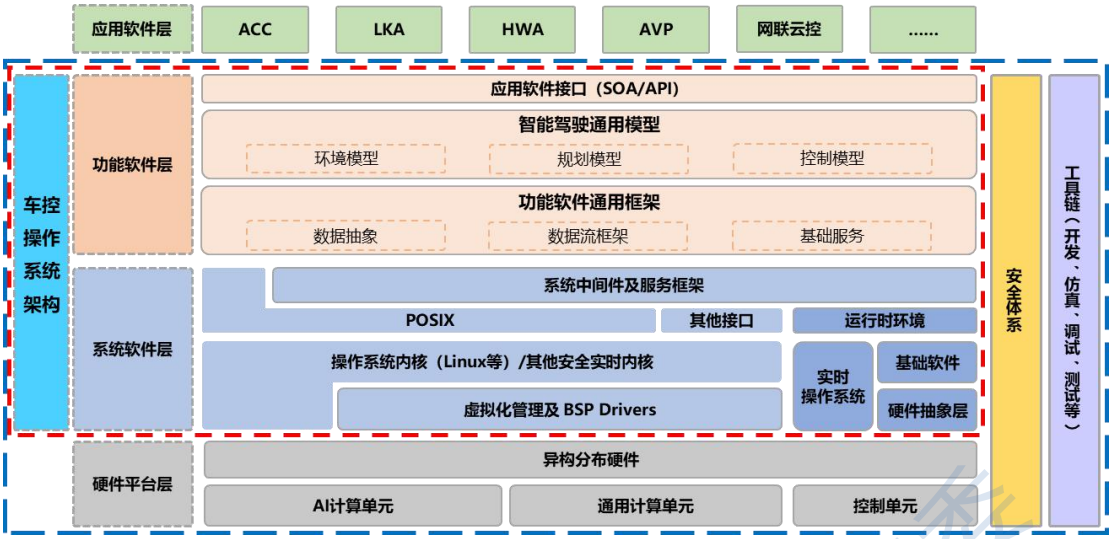

图 2　车控操作系统参考架构

（2）车载操作系统

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。

智能座舱通过多屏融合实现人机交互，以液晶仪表、HUD、中控屏及中控车载信息终端、后座 HMI 娱乐屏、车内外后视镜等为载体，实现语音控制、手势操作等更智能化的交互方式。未来有可能将人工智能、AR、ADAS、VR 等技术融入其中。车载操作系统是由不同的座舱电子组合成完整的体系，其向下集成芯片硬件，向上提供开发框架和算法库，支撑用户应用平台的定制开发，从用户端为语音和图像识别等多模人机交互、地图导航、V2X 网联通信、信息安全、“人-车-路-云”数据融合提供支撑。作为应用类程序的运行载体，车载操作系统发挥着构建智能网联汽车生态的作用，需要具备良好的移植性和便捷的开发接口。

《车载操作系统架构研究报告》及《车载操作系统总体技术要求

研究报告》也于2021年7月6日-8日，在第七届智能网联汽车技术及标准法规国际交流会（ICV2021）上正式发布，其中，《车载操作系统架构研究报告》重点介绍了车载操作系统总体架构、单系统及多系统架构和功能要求以及相关标准化建议；《车载操作系统总体技术要求研究报告》重点介绍了多系统技术要求、多核技术要求、基础服务技术要求、接口技术要求、安全技术要求以及相关标准化建议。

2.1 国内外车用操作系统发展现状

2.1.1 安全车控操作系统

安全车控操作系统主要面向车辆经典控制领域，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较高。目前安全车控操作系统使用比较多的有Free RTOS、MICROSAR OS、OSEK-OS等。

2.1.1.1 Free RTOS

Free RTOS是一种轻量级的开源实时操作系统，它于2003年由Richard Barry设计，目前已经成功移植到多种不同的微控制器上。该操作系统简单小巧，文件数量少，通常内核只占用4~9k的字节空间，且代码主要由C语言编写，可移植性高。不仅具有不同任务之间通信的方式，且具有优先级继承特性的互斥信号量，以及高效的软件定时器。

2.1.1.2 MICROSAR OS

MICROSAROS是Vector公司开发的一种抢占式实时多任务操作系统，完全符合AUTOSAR标准。Vector是全球领先的总线开发工具、

网络节点测试验证工具和嵌入式软件组件供应商，为汽车总线网络的设计、建模、仿真、分析、测试以及 ECU 的开发、测试、标定和诊断等领域提供一系列强有力的软硬件工具和源代码。MICROSAR OS 有如下优点：小型，快速，节省资源，启动时间短；可用作多核心操作系统；易于配置的图形配置工具；时间保护和内存保护的功能。

2.1.1.3 OSEK-OS

OSEK，是指德国的汽车电子类开放系统和对应接口标准（open systems and the corresponding interfaces for automotive electronics）。OSEK-OS 是个静态的操作系统，不支持在运行过程中动态更改，用户在产生特定的 kernel 之前，必须确定所需要资源的准确数目。另外，OSEK-OS 也不需要进行动态的内存管理。

OSEK 规范为实现其制定的初衷并满足汽车控制领域对系统安全性和节省有限资源的特殊要求，制定了系统而全面的操作系统规范，包括实时性，可移植性，可扩展性。

2.1.1.4 TEE Lite OS

2006 年，OMTP 工作组率先提出了一种双系统解决方案：即在同一个智能终端下，除了多媒体操作系统外再提供一个隔离的安全操作系统，这一运行在隔离的硬件之上的隔离安全操作系统用来专门处理敏感信息以保证信息的安全。该方案即 TEE 的前身。英国 ARM（Advanced RISC Machines）公司于 2006 年在 V6 指令集中提出 Trustzone 安全架构，以此支持 TEE 技术，并持续演进至 V9 架构中的全新 CCA 机密计算体系架构。由于 ARM 指令集芯片在车载芯片

中的占有率极高，因此 Trustzone 安全架构普遍存在于车控、车载芯片中。但是由于 Trustzone 需要部署专门的 TEE OS，因此目前绝大多数的车控、车载芯片中的 Trustzone 均处于未使用状态。

Trustzone 作为 ARM 的硬件安全架构，工作在芯片的特权层，且先于 Linux/Android 系统启动，因此可以为车控、车载操作系统及上层应用提供全面的可信安全能力支撑，但是由于车控芯片的算力限制以及业务的单一性，通常车控芯片 TEE OS 采用轻量级的 TEE Lite OS，主要用于升级过程的密钥证书的存储和完整性验证，以及芯片安全启动（Secure Boot）过程的可信验证。

2.1.2 智能驾驶操作系统

智能驾驶操作系统主要面向智能驾驶领域，用于智能驾驶辅助，以及全自动驾驶功能的 ECU 上。目前智能驾驶控制器主要使用的底层操作系统有 QNX 以及 Linux。

2.1.2.1 QNX

QNX 是一款微内核、嵌入式、非开源、安全实时的操作系统。QNX 是微内核架构，内核一般只有几十 KB，驱动程序、协议栈、文件系统、应用程序等都在微内核之外的受内存保护的空间内运行，可实现组件之间相互独立，避免因程序指针错误造成内核故障。因其内核小巧，运行速度极快，具有独特的微内核架构，安全和稳定性高，不易受病毒破坏系统，是全球首款通过 ISO 26262 ASIL-D 安全认证的实时操作系统。因此，QNX 常用于安全稳定性要求较高的数字仪表中。

QNX 凭借其安全、稳定等优势占据市场较高份额。QNX 为非开源系统，具有开发难度大、应用生态较弱等特点，而且需要商业收费。但由于现阶段汽车嵌入式操作系统对安全性、稳定性、实时性具有非常严苛的要求，QNX 凭借这些优点仍牢牢占据汽车嵌入式操作系统市占率第一的位置。根据官网显示，QNX 已与 45 家以上 OEM 进行合作，超过 1.75 亿辆汽车使用了 QNX 系统。

2.1.2.2 Linux

Linux 是一款开源、功能更强大的操作系统。Linux 具有内核紧凑高效等特点，可以充分发挥硬件的性能。它与 QNX 相比最大优势在于开源以及更丰富的生态，具有很强的定制开发灵活度。基于 Linux 开发新的操作系统是指基于 Linux Kernel 进一步集成中间件、桌面环境和部分应用软件。Linux 功能较 QNX 更强大，组件也更为复杂，因此 Linux 常用于支持更多应用和接口的信息娱乐系统中。

2.1.2.3 特斯拉 OS

特斯拉的操作系统 Version 基于 Linux 内核深度改造而成。特斯拉系统平台采用 Linux4.4 开源操作系统（Github.com 2021 年 5 月最新版本更新至 4.14），支持 PyTorch 的深度学习编程框架，基于 Kafka 开源流实时数据处理平台，可支持信息娱乐系统（IVI）和驾驶辅助系统（ADAS）等。

对于信息安全问题，特斯拉使用了 Linux 系统中的内核模块：安全增强型 Linux（SELinux），通过“访问权限控制”增加了操作系统信息安全性。访问权限控制，是指了解系统内所有的硬件资源、设

备接口明确允许访问的范围和硬件接口。简单来说，即为第三方软件划分可访问与禁止访问区域，最大限度地保证自身安全。通过访问权限控制，即便第三方程序对操作系统进行了攻击，特斯拉也可以保证核心区域不受破坏。

2.1.2.4 Harmony OS

2019 年 8 月，华为推出鸿蒙 Harmony OS，这是一款基于微内核的全场景分布式 OS，其开发的初衷是为了提升操作系统的跨平台能力，包括支持全场景、跨多设备和平台以及应对低时延和高安全性挑战的能力。鸿蒙系统具有四大特点：分布架构、天生流畅、内核安全和生态共享；有三层架构：第一层是内核，第二层是基础服务，第三层是程序框架。2019 年鸿蒙 OS 1.0 率先用于智慧屏产品，计划从 2020 年起将逐步用于手机、平板、汽车等更多智能设备中。

2020 年，华为自动驾驶操作系统内核获得业界 Safety 领域最高等级功能安全认证（ISO 26262 ASIL-D），成为我国首个获得 ASIL-D 认证的操作系统内核；同时，该内核于 2019 年 9 月获得 Security 领域高等级信息安全认证（CC EAL 5+），标志着该系统内核已成为业界首个拥有 Security & Safety 双高认证的商用 OS 内核。

2.1.2.5 百度 OS

百度的操作系统 Apollo 是一个开源的自动驾驶平台，旨在向汽车行业提供一个开放、完整、安全的软件平台，帮助他们结合车辆和硬件系统，快速搭建一套属于自己完整的自动驾驶系统。百度将开放环境感知算法、路径规划算法、车辆控制算法、车载操作系统的源代

码，并提供完整的开发测试工具，联合市场上成熟的传感器等领域合作伙伴，一同致力于降低无人车的研发门槛。

2.1.2.6 须弥 OS

须弥的操作系统 Hypervisor 是一款工作于智能驾驶系统底层的微内核虚拟机管理系统，通过 ARM 芯片的硬件虚拟化技术与 EL2 特权机制实现上层虚拟机资源调度与安全隔离。须弥 Hypervisor 自身并不单独作为智能驾驶操作系统使用，用户可以根据智能驾驶的业务需求在须弥 Hypervisor 中构架不同的虚拟机以运行 QNX 或 Linux 等智能驾驶操作系统，这种架构的好处在于：

（1）车辆制造商可以在一套智能驾驶零部件中同时适配和使用多个平台的智能驾驶 OS，从而防止单一供应商产生的供应链风险；

（2）车辆制造商可以利用 Hypervisor 提升功能安全 ASIL 等级，比如通过在两个虚拟机中同时运行两套 ASIL B 等级的智能驾驶系统，从而实现 ASIL C 等级的功能安全要求；

（3）车辆制造商可以利用虚拟机对深度学习、融合感知、行为预测、路径规划等算法进行隔离保护，确保核心知识产权的机密性和完整性，尤其是对环境感知过程中的人脸、车牌号等个人敏感信息的存储、处理以及脱敏活动进行保护，从而满足车辆网络安全相关标准以及个人数据保护法等法规的要求。

此外，由于须弥 Hypervisor 工作在芯片的特权层，且先于智能驾驶操作系统启动，因此可以与 TEE OS 相配合为智能驾驶操作系统及上层应用提供全面的可信安全能力支撑，如升级过程的密钥证书存储

和验证，以及基于芯片安全启动（Secure Boot）的信任链管理。

2.1.2.7 V-Trust TEE OS

2006 年，OMTP 工作组率先提出了一种双系统解决方案：即在同一个智能终端下，除了多媒体操作系统外再提供一个隔离的安全操作系统，这一运行在隔离的硬件之上的隔离安全操作系统用来专门处理敏感信息以保证信息的安全。V-Trust TEE OS 是谦川科技基于安全微内核与 Secure SOA 架构设计开发的二代 TEE OS。V-Trust 支持多核多线程，可以完全发挥智能驾驶芯片的高性能运算能力，确保智驾系统在执行安全操作时不会发生性能瓶颈。同时 V-Trust 基于 Secure SOA 架构将网络安全能力与核心逻辑功能抽象为标准服务接口，不仅可以为芯片内部的智能驾驶操作系统提供安全服务，同时还可以通过 SOMEIP 等面向服务的通信协议为车载总线上的其他零部件提供安全服务。Secure SOA 作为一种可扩展的安全框架，可以结合车辆制造商的业务需求、网络安全合规需求、数据与隐私保护需求进行迭代更新，并利用 V-Trust 的远程更新能力实现 Secure OTA。

2.1.3 车载操作系统

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娱乐系统，其中特斯拉和百度 Apollo 主要是基于 Linux 和 QNX 系统定制开发的，同时也可以属于车载操作系统范畴，在智能驾驶操作系统已经描述，这里不再说明，目前各个 OEM 以及 Tier1 使用的主要是 Android 以及使用 Android 定制开发的，详细介绍如下。

2.1.3.1 Android Automotive

谷歌于 2019 年发布 Android Automotive OS，是一款可直接运行在汽车 IVI 系统上的开源操作系统，用户可以通过 Google Play 下载 Google 助手、Google Map 等应用在汽车上运行，而无需使用 Android 手机。Android Automotive 与手机 Android 类似，其源代码库免费和开源，提供基本的信息娱乐功能，主机厂可通过 Android 的通用框架和 API 来实现自己所需的功能。

Android Automative 是在原手机 Android 的系统架构基础上替换为与车相关的模块。主要包括：1）Car APP：包括 OEM 和第三方开发的 APP；2）Car API 提供给汽车 APP 特有的接口；3）Car Service：系统中与车相关的服务；4）Vehicle Network Service：汽车的网络服务；5）Vehicle HAL：汽车的硬件抽象层描述。区别于之前的开源安卓系统，车载安卓系统的灵活可定制性和可修改编辑性大大降低，其应用或许受限。

2.1.3.2 AliOS

2015 年，阿里基于 Linux Kernel LTS 开发出 YunOS，并与上汽合资成立斑马网络公司进行业务推广。YunOS 在 2017 年更名为 Alios，并在 2017 年宣布开源。2018 年，阿里发布 AliOS 2.0 系统。AliOS 属于智能座舱 OS，不包含自动驾驶功能，它通过先进的交互方式对座舱内部件的操控以及获取娱乐导航等信息服务，通过环境场景的感知来为用户提供驾驶辅助和服务推荐，可以实现车辆近远程控制（借助手机 APP，车主远离车辆也可以进行上锁、解锁、打开后备箱、打

开空调系统和座椅加热）和车辆状态查询。

AliOS 添加了云服务相关的模块以接入阿里巴巴的生态环境，重点包括电子商务（淘宝）、网络支付（支付宝）和高清地图（高德），打造手机之外的第二移动支付终端。在开发方面，AliOS 提供了基于云的高效开发框架 CAF，开发者只需要学习一次就可以在多个端口上进行开发，可同时支持手机、平板、互联网汽车、智能家居等设备，实现多端统一。前已有上汽荣威、上汽名爵、上汽大通、上汽大众斯柯达、宝骏、神龙、东风雪铁龙、福特、观致等使用了基于 AliOS 的斑马系统。

2.1.3.3 蘑菇 OS

蘑菇 OS 由蘑菇车联自主研发，是专用于车载场景的操作系统，具有多硬件平台通用适配能力，可按需扩展，快速构建定制化解决方案，特点是全面适配主流品牌芯片，并具有高开放性、高稳定性及高安全性。在多任务并行处理过程中，蘑菇 OS 利用多核心、高主频的异构计算处理器配合多线程并发，辅以异构异调进程调度策略，有效分配车机硬件资源，实现多任务协同管理，并控制各项任务优先级别，极大提高了系统稳定性。数据安全上，蘑菇 OS 从云、管、端整个数据链路着手，在数据源头和数据存储上打造了一套全流程、广纵深的立体化动态数据安全防护体系。蘑菇 OS 搭载自研的智能语音引擎，支持车辆控制、内容服务、安全预警等 10 大类 100+场景化语音交互，识别准确率达到 97%以上，能够在驾驶中实现全场景语音交互，大幅降低事故发生概率。

2.1.3.4 博泰 OS

博泰车联网擎OS是最早基于Android深度定制的车载底层架构。在保留 Android 开源特性的同时，擎 OS 基于软件与芯片级方案来增加系统的安全性能，且定制了与 QNX、Linux 操作系统的对接接口，允许车企根据自身需求实现多样化定制。操作层面，擎 OS 打破了传统操控的繁琐程序，真正做到全场景语音覆盖，为用户提供千人千面的个性化服务，了解用户的真实需求；在服务层面，擎 OS 打通了车辆数据、用户数据与云端接口，为生态开发奠定了基础。

2.1.3.5 蔚来 OS

蔚来汽车 NIO OS 致力于创造愉悦的驾乘体验的数字座舱。期间与用户深度共创，敏捷迭代，结合蔚来用户的实际体验反馈，不仅使系统的功能更丰富、人性化更高，更高效操作的特点确保用户的安全。另外，通过模拟驾驶实验、眼动测试、心率测试等十多种专业测试，界面在充满视觉美感的同时也兼具了工程严谨性，让驾驶者在驾驶或者停车过程中都能够轻松操作各种功能。NIO OS 内置的媒体应用让用户自由选择音乐、FM、有声书，以及蔚来专属的 NIO Radio，集成化的桌面地图导航应用、爱车应用、可自定义的快捷控制、互联账户系统以及常用常新的可持续升级能力，让用车人的车载生活越来越丰富。

2.1.3.6 小鹏 OS

小鹏 Xmart OS 车载智能系统基于安卓开发，可以快速适配大量安卓平台已有的应用，拥有 AI 语音助手、远程 APP 操控、整车 OTA

升级服务、智能导航、音乐与有声读物和车载应用商城等功能。除了常规的车机功能外，Xmart OS 主要从远程控制、AI 智能以及不断拓展的 V2X 几方面来深度展示其出色的智能优势。

2.1.3.7 比亚迪 OS

DiLink 智能网联系统是比亚迪基于智能硬件、车内网、云端通讯、信息融合、AI、大数据等技术和用户洞察，完全独立自主研发的技术+内容的服务生态体系。DiLink 智能网联系统包含 Di 平台、Di 云、Di 生态和 Di 开放四大能力平台。

2020 年 8 月，比亚迪正式发布了 DiLink3.0 系统，最大亮点是新加入了平台 DiUI，升级为五大能力平台，并加入了一些新功能，同时在系统软硬件等方面做了升级。除了新平台 DiUI 的加入以外，DiLink 原有四大平台上也得到了升级，比如千里眼、手机 NFC 车钥匙、高温杀毒、语音功能、车内 K 歌等功能。

2.1.3.8 吉利 OS

吉客智能生态系统（Geek User Interaction, GKUI）是吉利汽车基于安卓平台打造的汽车智能生态系统，该系统由“一朵云”、“一个桌面”、“一个 ID”以及“应用生态”四大部分组成。

2019 年，GKUI 19 版在 2018 年版的基础上，结合 GKUI 用户的生活方式和用车习惯，搭载性能更强的硬件平台。“一朵云”升级为“AI 云（AI Cloud）”，大幅增强用户端人机交互的智能应用体验。“一个桌面”升级为“智能桌面”实现了控件交互，场景推荐等功能。“一个 ID”（相当于车主的“身份证”）则升级成“ID 生态”，用

来打通更多互联场景。“生态系统”升级为互联网生态，全面融入更丰富的互联网生态服务，并以“生态开放”为指引，为小米、百度等第三方开发者提供更广阔的服务平台。

2.1.3.9 须弥 OS

须弥 Hypervisor 不仅可以工作在智能驾驶操作系统底层（见2.1.2.6），还可以为车载操作系统提供底层的微内核虚拟机管理服务，通过ARM芯片的硬件虚拟化技术与EL2特权机制实现上层虚拟机资源调度与安全隔离。须弥 Hypervisor 自身并不单独作为车载操作系统使用，用户可以根据车联网业务需求在须弥 Hypervisor 中构架不同的虚拟机以运行 Linux 或 Android 等车载操作系统，这种架构的好处在于：

（1）车辆制造商可以在一套智能座舱中同时适配和使用多个平台的车载操作系统，以支持不同的业务，比较常见的方案是，利用 Linux 和 Android 两个虚拟机分别运行仪表盘和车机娱乐系统；

（2）车辆制造商可以利用 Hypervisor 提升功能安全 ASIL 等级，比如通过在两个虚拟机中同时运行两套 ASIL B 等级的智能驾驶系统，从而实现 ASIL C 等级的功能安全要求，或通过须弥 Hypervisor 内核监控仪表盘系统状态，当发生异常时，快速重启仪表盘系统；

（3）车辆制造商可以利用虚拟机对深度学习、身份验证、行为预测、驾驶员状态识别等算法进行隔离保护，确保核心知识产权的机密性和完整性，尤其是对驾驶员监控系统或行车记录仪中涉及到的人脸、车牌号等个人敏感信息的存储、处理以及脱敏活动进行保护，从

而满足车辆网络安全相关标准以及个人数据保护法等法规的要求。

此外，由于须弥 Hypervisor 工作在芯片的特权层，且先于车载操作系统启动，因此可以与 TEE OS 相配合为车载操作系统及上层应用提供全面的可信安全能力支撑，如升级过程的密钥证书存储和验证，以及基于芯片安全启动（Secure Boot）的信任链管理。

2.1.3.10 V-Trust TEE OS

V-Trust TEE OS 是谦川科技基于安全微内核与 Secure SOA 架构设计开发的二代 TEE OS。V-Trust 支持多核多线程，可以完全发挥智能座舱芯片的高性能运算能力，确保车载操作系统在执行关键操作或安全任务时不会发生性能瓶颈。同时 V-Trust 通过 Secure SOA 架构将网络安全能力与核心逻辑功能抽象为标准服务接口，不仅可以为芯片内部的车载操作系统提供安全服务，还可以通过 SOMEIP 等面向服务的通信协议为车载总线上的其他零部件提供安全服务。Secure SOA 作为一种可扩展的安全框架，可以结合车辆制造商的业务需求、网络安全合规需求、数据与隐私保护需求进行迭代更新，并利用 V-Trust 的远程更新能力实现 Secure OTA。V-Trust TEE OS 大致可实现如下三类功能，并以 Secure SOA 的形式提供相关安全服务：

（1）针对关键数据的隔离及安全存储，如信任根、证书密钥与核心业务逻辑；

（2）针对关键业务逻辑的执行保护，例如加解密算法、访问控制策略、身份验证过程、车内无感支付及其他核心商业逻辑代码；

（3）针对个人隐私数据的保护及匿名化处理，例如生物识别信

息、驾驶行为数据、地理位置信息等。

2.1.3.11 WinCE

WinCE 是微软 1996 年发布的嵌入式操作系统，主要应用于车载主机、车载导航和车载娱乐系统。但是随着 Linux 和 Android 的冲击，现阶段开发者和应用者已非常少了，微软计划于 2021 年 3 月终止对其服务，将逐步退出汽车操作系统市场。

2.2 操作系统测试需求分析

由于智能网联汽车产业发展迅速，产品迭代速度加快，随着软件定义汽车概念的兴起，整车企业出于构建软件生态的需要，不断推出新的汽车软件平台，汽车操作系统呈现出百花齐放的发展态势。作为安全性和可靠性要求极高的智能网联汽车系统，迫切需要统一的操作系统技术和测试标准体系，以保障产品的质量和消费者权益。

在“核高基”支持下，我国在汽车嵌入式实时操作系统领域取得了一定进展，但由于技术产品基础薄弱、专业人才不足、生态支撑能力差、市场拓展困难等因素，导致我国在传统车控操作系统领域话语权仍然不强。近年来，我国在智能车控、车载操作系统领域涌现出了华为、百度、阿里等有实力的企业，但我国车用操作系统发展仍处于初期阶段，在智能车控、车载关键核心技术研发、软硬件兼容适配、测试评价等方面仍需强化，生态支撑能力有待加强。且由于国内相关标准法规的不健全，不论是主机厂还是 Tier1，或者其他参与方，在车用操作系统测试领域都没有一套可以遵循的标准，整个生态都依赖零部件功能测试，间接保证底层操作系统的可用性。由于测试标准的

欠缺，导致上下游合作时，验收标准定义模糊、后期出现验收困难、以及责任不清的情况。另外，我国在车控、车载关键核心标准制定、测试技术、应用软件以及配套硬件供给方面与国外仍存在一定差距。企业间、企业与科研机构间的技术合作、资本合作等合作模式有待探索，产学研用协同创新体系尚未形成。

因此，亟需根据中国实际情况制定一套中国车用操作系统的测评标准，对相关技术进行量化测评，从而准确把控风险，同时促进成熟技术落地，推动整个汽车底层软件生态更加健康规范地发展。

2.3 车用操作系统测试研究现状

车用操作系统是用于控制和管理汽车硬件与软件资源的程序，是保障汽车正常运行和功能性能的基础和关键。在此形势下，对车用操作系统的测试是保证车辆行车安全的重中之重。通过对车用操作系统的技术测试来发现系统存在问题，能有效减少操作系统本身的问题对车辆行车安全造成的威胁。因此本研究对车用操作系统的测试方法和技术具有重要的意义。国内外研究人员已经对操作系统的测试进行了相关研究，但目前还未形成系统成熟的测试方法。下面就国内外对车用操作系统的测试内容和测试方法进行分析。

目前车用操作系统的测试主要指基本功能组件测试，涉及的测试类型包括：文档审查、代码审查、功能测试、性能测试、接口测试、余量测试、边界测试、人机交互界面测试、强度测试、安全性测试、恢复性测试、标准符合性测试、安装性测试等。操作系统向下兼容不同的硬件平台，向上对应用层提供相对统一的接口支持。

由于目前各厂商产品缺乏统一的产品研发规范和标准，经常出现由于操作系统版本的变化导致硬件设备无法兼容使用的情况，因此对操作系统进行硬件的兼容性测试就显得尤为重要，大多操作系统的测试方法是在国产化关键软硬件测试中，对硬件兼容性做大量的适配测试。

目前专业操作系统厂商具备一定的适配硬件资源，即便如此，缺少兼容性检测技术规范以及手段且工作量巨大，导致由厂商接入硬件直接开展兼容性测试变得尤为困难。而目前国内外针对操作系统兼容性测试的研究主要集中在软件方面，目前对国产操作系统的硬件兼容性测试还没有一种通用的方法，针对不同的底层硬件，使用不同的测试工具或编写脚本代码进行测试。

操作系统向上提供应用层开发标准接口，标准接口的目的是统一的环境来支持源代码级的可移植性，提高不同操作系统的兼容性和应用程序的可移植性而制定的一套标准。操作系统厂商针对开放接口的测试基本采用打桩单元测试的方法进行接口测试，以保证接口的一致性和稳定性。部分接口为内部接口，不对外开放，不对一致性进行测试。测试完成，以 SDK 形式对外释放接口使用说明，但测试方法以及测试结构一般仅供内部质量管控跟踪使用，不对外开放。

操作系统标准接口符合性测试主要包括：

（1）POSIX 标准符合性测试，操作系统的系统调用与编程接口支持 POSIX 规范的系统调用接口，保障应用的可移植性。具体包括：POSIX_DEVICE_IO、POSIX_NETWORKING 等类。

（2）TCP/IP 标准符合性测试，验证 BSD SOCKET 套接字的标准函数接口。

字符集标准符合性测试，验证国产操作系统数据及访问接口是否支持 GBK、ASCII、GB18030、UNICODE，并验证在操作系统文件编码不一致时数据访问的正确性。

随着操作系统在关键场景中应用，非功能性测试与功能性测试同样重要，测试体系也发展起来。

（1）负载测试

在这里，负载测试指的是最常见的，验证一般性能需求，而进行的性能测试。因此负载测试主要是考察操作系统在既定负载下的性能表现。

（2）压力测试

压力测试是为了考察系统在极端条件下的表现，这个极端条件并不一定是用户的性能需求，可能要远远高于用户的性能需求。可以这样理解，压力测试和负载测试不同的是，压力测试的预期结果就是系统出现问题，而我们要考察的是系统处理问题的方式。比如说，我们期待一个系统在面临压力的情况下能够保持稳定，处理速度可以变慢，但不能系统崩溃。因此，压力测试是能让我们识别系统的弱点和在极限负载下程序将如何运行。压力测试包括内存管理压力测试、文件系统压力测试、数学（浮点）测试、多线程压力测试、硬盘 I/O 测试、IPC（pipeio, semaphore）测试、系统调用功能的验证测试、网络压力测试等。

（3）基准测试

当系统中增加一个新的模块的时候，需要做基准测试，以判断新模块对整个软件系统的性能影响。按照基准测试的方法，需要打开/关闭新模块至少各做一次测试。关闭模块之前的系统各个性能指标记下来作为基准（Benchmark），然后与打开模块状态下的系统性能指标作比较，以判断模块对系统性能的影响。

（4）稳定性测试

测试系统在一定负载下长时间运行后是否会发生问题。软件系统的有些问题是不能一下子就暴露出来的，或者说是需要时间积累才能达到能够度量的程度。如，内存泄漏问题。

（5）可恢复测试

操作系统能否快速地从错误状态中恢复到正常状态。如，操作系统承受了压力无法正常工作后，是否可以记录出错信息，并且是否能够快速恢复到正常状态，可恢复测试通常结合压力测试一起来做。

2.4 国内外车用操作系统行业规范/标准现状

2.4.1 国外车用操作系统行业规范/标准现状

车用操作系统在国外发展较早，目前已经开展了一系列的标准化工作。

2.4.1.1 AUTOSAR

欧洲在 20 世纪 90 年代发展出用于汽车电子领域，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（OS）；2）通信；3）网络管理；4）OSEK 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 3 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 3　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、横向模块化架构以及代码开源方式，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 家的车企、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。

2.4.1.2 POSIX

POSIX，是 IEEE 为在各种类 UNIX 操作系统上运行软件，而定义 API 的一系列互相关联的标准的总称，其正式称呼为 IEEE Std 1003，而国际标准名称为 ISO/IEC 9945。此标准源于一个大约开始于 1985 年的项目。POSIX 这个名称是由理查德·斯托曼（RMS）应 IEEE 的要求而提议的一个易于记忆的名称。它基本上是 Portable Operating System Interface（可移植操作系统接口）的缩写，而 X 则表明其对 Unix API 的传承。该 POSIX 规范在 1988 年释放，目前最新的版本为 IEEE Std 1003.1-2017。上述智能驾驶操作系统 Linux 和 QNX 以及车载操作系统都遵守 POSIX 规范实现。

2.4.1.3 AGL 规范

2014 年，Linux 基金会发布了开源 AGL（Automotive Grade Linux）规范 1.0 版本，它是业界首个开放式车载信息娱乐（IVI）软件规范，

而且新的规范发布对细节进行更精确的规范，同时更好的满足任何遵循 AGL 规范的 IVI 栈定义汽车级 Linux 软件平台的体系结构。该 AGL 规范对 APP/HMI Layer、Application Framework Layer、Services Layer、Security Services、Operating System Layer 等方面进行了相关的规范说明，AGL 架构如图 4 所示：

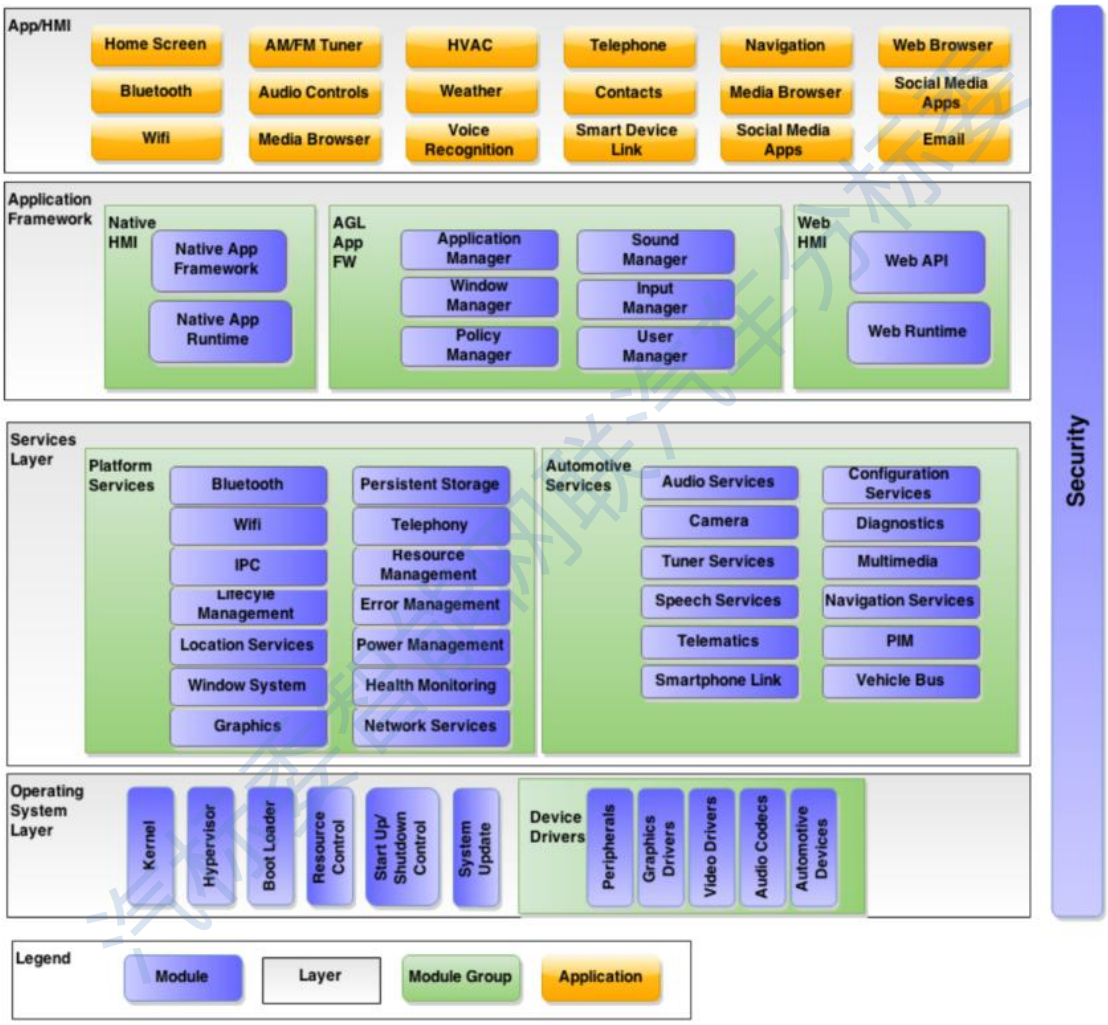

图 4　AGL 架构图

2.4.1.4 IEC 61508

IEC 61508 完整的名称是《电气/电子/可编程电子安全相关系统的功能安全》，它针对采用了硬件、软件、电子、电气、机械等多种技术的综合系统，提出了端到端、全系统和全生命周期的安全评估理

念，意在解决科技发展中越来越复杂的工程系统带来系统失效模式和失效率预测困难的问题。

所谓端到端的概念，就是明确安全要素的目标与当前能力水平的一种思路。标准提出了安全性等级的概念，把系统和它的每一级子系统进行标准评估，指明每个层级组成元素的安全指标和实际能力与安全指标的距离，得出每一个评估主体的具体安全等级作为标签。给每个安全因素分配明确的努力方向，并逐级分解，这个思路使得每个底层产品的供应商都有了自己的具体目标，避免了目标不清带来的失效。有研究提出了一个数字：40%，研究认为40%的失效都是因为安全要求不清晰造成的。

全系统理念，是在端到端理念的基础上，要求考虑全系统的结构、逻辑等系统才能体现出来的性质对安全的影响，要求除了明确每个零件的具体参数指标、可靠性指标之外，这些零件组合以后产生的新的性质，也必须得到考察指定安全目标，衡量安全级别。

全生命周期理念，一般的性能测试或者质量检查，都是针对某一个时刻的产品状态，是整个生命周期中的一个切片。IEC 61508 要求系统考虑产品系统从概念到设计直至使用以后的寿命终结各个阶段，考虑时间因素在产品安全中产生的影响。

2.4.1.5 ISO 26262

ISO 26262《道路车辆功能安全》是从电子、电气及可编程器件功能安全基本标准 IEC 61508 派生出来的，主要定位在汽车行业中特定的电气器件、电子设备、可编程电子器件等专门用于汽车领域的部

件，旨在提高汽车电子、电气产品功能安全的国际标准。

ISO 26262 从 2005 年 11 月起正式开始制定，经历了大约 6 年左右的时间，已于 2011 年 11 月正式颁布，成为国际标准。2018 年 12 月，ISO 26262 标准的第二个版本发布，更新了卡车、公共汽车、摩托车等相关要求。

ISO 26262 为汽车安全提供了一个生命周期（管理、开发、生产、经营、服务、报废）理念，并在这些生命周期阶段中提供必要的支持。该标准涵盖功能性安全方面的整体开发过程（包括需求规划、设计、实施、集成、验证、确认和配置）。

ISO 26262 标准根据安全风险程度对系统或系统某组成部分确定划分由 A 到 D 的安全需求等级（Automotive Safety Integrity Level 汽车安全完整性等级 ASIL），其中 D 级为最高等级，需要最苛刻的安全需求。伴随着 ASIL 等级的增加，针对系统硬件和软件开发流程的要求也随之增强。对系统供应商而言，除了需要满足现有的高质量要求外还必须满足这些因为安全等级增加而提出的更高的要求。

2.4.1.6 MISRA C

MISRA 全称 Motor Industry Software Reliability Association（汽车工业软件可靠性协会），是由汽车制造商、零部件供应商、工程咨询师代表组成的联盟，旨在推动开发安全相关的嵌入式软件在车辆及其他嵌入式系统中的最佳实践。MISRA-C 是由该联盟提出的 C 语言开发标准，其目的是增进 C 代码在嵌入式系统中的安全性、可移植性和可靠性。针对 C++语言也有对应的 MISRA C++。

MISRA-C 一开始主要是针对汽车产业，不过其他产业也逐渐开始使用 MISRA-C，包括：航天、电信、国防、医疗设备、铁路等领域中都已有厂商使用。

2.4.1.7 ASPICE

ASPICE 全称是“Automotive Software Process Improvement and Capacity Determination”，汽车软件过程改进及能力评定，是汽车行业用于评价软件开发团队的研发能力水平的模型框架。最初由欧洲 20 多家主要汽车制造商共同制定，于 2005 年发布，目的是为了指导汽车零部件研发厂商的软件开发流程，从而改善车载软件的质量。ASPICE 从最开始的 2.0 版本，不断发展更新，现在最新的 ASPICE 是 2017 年 11 月发布的 3.1 版本。

多年以来，ASPICE 在欧洲汽车行业内被广泛用于研发流程改善及供应商的研发能力评价。随着近年车联网、智能驾驶、新能源汽车的迅速发展，软件在汽车研发中的占比激增，企业对软件质量管理的需求不断增强，ASPICE 逐渐被引入到国内，被国内的企业所熟知。另一方面随着 TS16949 的改版，对企业提出定期审核的要求，也对 ASPICE 在国内的应用起到的极大的促进作用。

2.4.2 国内车用操作系统行业规范/标准现状

国内目前操作系统行业规范/标准制定主要处于跟随状态，国内主机厂及零部件供应商多使用 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（中国汽车电子基础软件自主研发与产业化联盟，China Automotive Electronics Industry Infrastructure Software Alliance），旨在中国推广和发展 AUTOSAR 架构。

国内加速布局车用操作系统相关标准，2019 年 10 月 25 日全国汽车标准化技术委员会（以下简称“汽标委”）在世界智能网联汽车大会正式发布《车用操作系统标准体系》，2021 年 7 月 6 日-8 日，汽标委在第七届智能网联汽车技术及标准法规国际交流会（ICV2021）上正式对外发布了十项“2021 年度智能网联汽车标准化需求研究成果”，包括《车控操作系统架构研究报告》、《车控操作系统总体技术要求研究报告》、《车载操作系统架构研究报告》、《车载操作系统总体技术要求研究报告》等，报告中提出了车用操作系统标准化建议，对未来国内车用操作系统的制定提供了指导。

此外，汽车操作系统相关团体标准也在加速布局，由中国智能网联汽车产业创新联盟（CAICV）提出，国汽智控（北京）科技有限公司牵头编制的《车控操作系统功能软件架构及接口要求》CSAE 标准已按《中国汽车工程学会标准（CSAE）制修订管理办法》有关规定通过立项审查，现正式列入中国汽车工程学会标准研制计划，起草任务书编号为：2021-14。该标准从总体架构、总体要求、接口要求和安全要求等方面对车控操作系统功能软件的技术要求进行规范，为功能软件的设计开发提供指导。

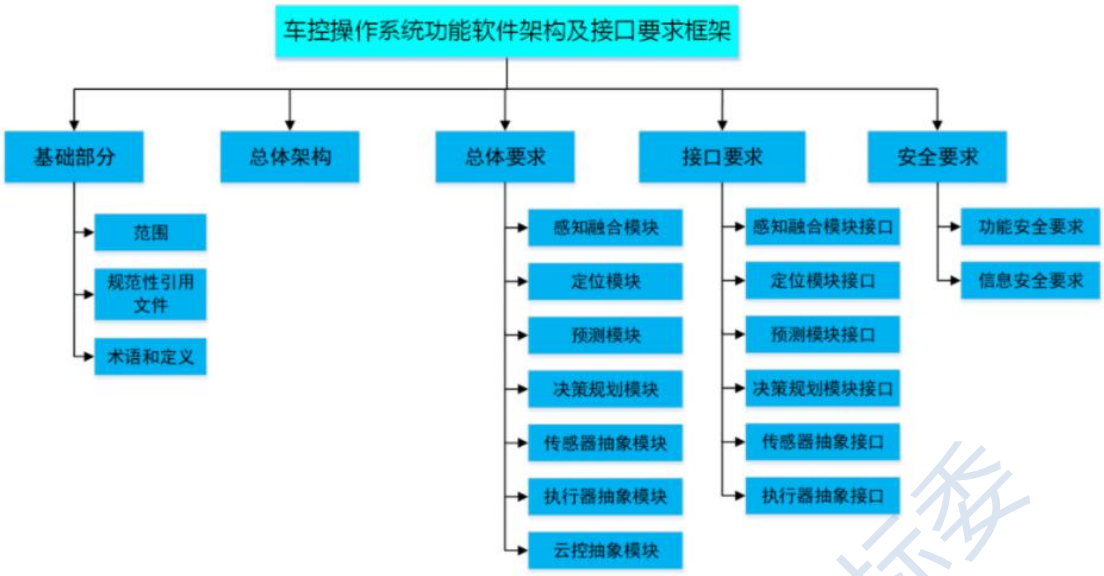

图 5　CSAE 标准《车控操作系统功能软件架构及接口要求》框架

在 TEE 可信执行环境操作系统方面，相关标准的进展发展较快，并在金融行业的成熟最佳实践和行业标准的指导下，已经形成了相对完善的国家标准《信息安全技术 可信执行环境 基本安全规范》，并预计将于 2021 年底正式发布。该标准由中国银联、信通院、华为、小米以及谦川科技等多家机构共同编制完成。

该标准确立了可信执行环境系统整体技术架构、硬件要求、安全启动过程基本要求、可信虚拟化、 可信操作系统、可信应用与服务管理基本要求、可信服务基本功能及要求、跨平台应用中间件、可信应用架构及安全要求、测试评价方法的相关技术要求。可用于指导可信执行环境系统的设计、生产及测试。并可进一步转化为汽车行业标准，纳入车用操作系统标准体系中。

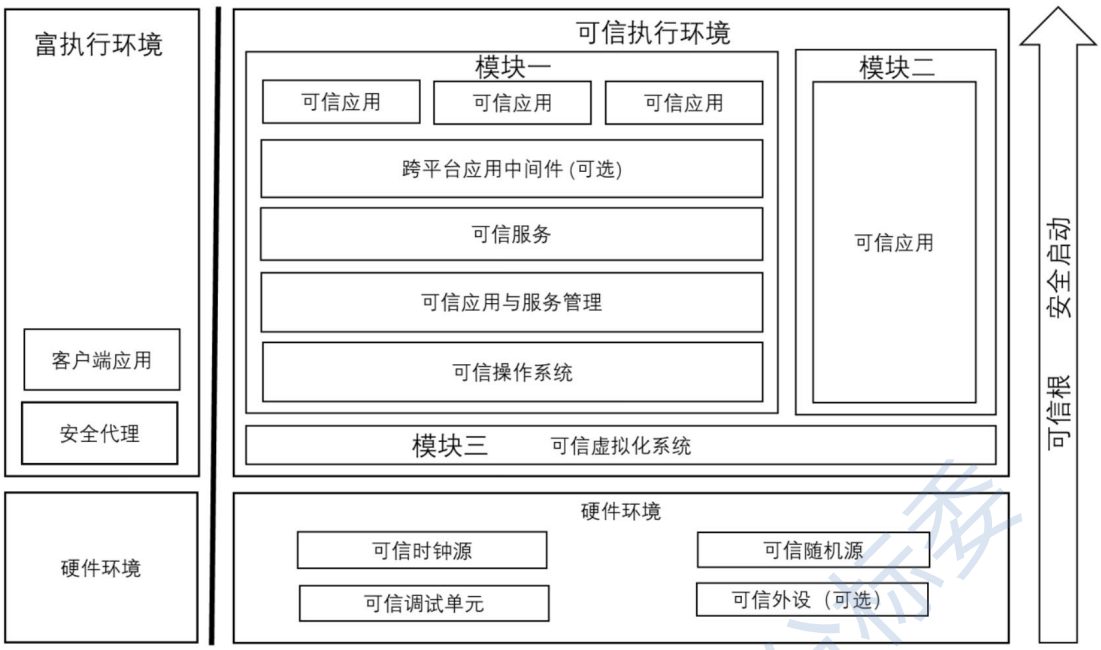

图 6　国标《信息安全 可信执行环境 基本安全规范》整体框架

3　车用操作系统测试研究基础

3.1 操作系统通用测试概述

3.1.1　操作系统通用测试方法

操作系统是由系统软件和功能软件组成的程序集合。在进行车用操作系统测试时，要在规定的条件下对相应的程序进行有效操作，发现程序中存在的错误，并对操作系统自身品质进行分析，从而评估软件自身品质是否满足设计要求。车用操作系统测试是整个操作系统开发中必不可少的重要环节，操作系统是产品运行的核心软件，是影响产品自身品质的关键因素。为此，要加强操作系统测试基本测试方法的分析。

3.1.1.1 黑盒、白盒测试

在对软件测试时，可以把程序当作一个不能打开的黑盒子，在对程序内部结构以及内部特性不进行全方位的考虑下，对其程序接口进

行相应测试，检查程序自身功能是否依照需求说明书规定进行正常有效的使用，以用户角度出发，根据产品自身应该具备的实际功能和定义完成的产品规格，对产品自身应该具备的功能进行检验，要保障每个功能都可以正常应用，并且满足客户需求。白盒测试又称透明盒测试、结构测试等，是软件测试的主要方法之一。白盒测试用来测试应用程序的内部结构或运作，而不是测试应用程序的功能（即黑盒测试）。在白盒测试时，以编程语言的角度来设计测试案例。测试者输入资料，验证资料流在程序中的流动路径，并确定适当的输出，类似测试电路中的节点。测试者了解待测试程序的内部结构、算法等信息，这是从程序设计者的角度对程序进行的测试。黑盒测试无法取代白盒测试，它与白盒测试属于互补的测试方法，可以将白盒测试中不易发现的其他类型错误有效的展现出来。白盒测试主要对程序代码逻辑进行有效测试，而黑盒测试是程序所展现给用户的功能，白盒测试属于软件自身后台程序。

3.1.1.2 静态、动态测试

动态测试方法是指通过运行被测程序，检查运行结果与预期结果的差异，并分析运行效率、正确性和健壮性等性能。这种方法由三部分组成：构造测试用例、执行程序、分析程序的输出结果。静态测试是指不运行被测程序本身，通过分析或检查源程序的语法、结构、过程、接口等来检查程序的正确性。在应用动态测试时，需要应用相应的运行软件对系统的动态行为进行有效测试，并以动态的工作对其进行相应的测试分析，在动态测试中包含有测试用例以及测试程度，而

应用范围则包含有单元检测以及集成测试。静态检测会对相应的源程序以及数据定义进行控制检测，通过静态测试可以对相应的代码进行审查以及静态分析，在进行代码审查时，需要由人工对其进行检测，并对相应的代码进行评审，从而有效发现代码中存有的缺陷。

3.1.1.3 单元测试

单元测试是指依据详细的设计描述，对每一个功能相对独立的程序模块进行测试，检查各个单元是否正确地实现规定的功能。单元测试指独立测试系统中的每个独立单元(函数)，在该阶段，主要是测试单独的函数，而不关心该函数所调用的其他函数的运行情况。单元测试通常由软件开发人员编写，用于确保他们所写的代码符合软件需求和遵循开发目标。一般单元测试采用的做法是在开始阶段选用黑盒测试，根据输入输出测试函数的接口，然后再用白盒测试技术对函数代码的执行路径进行分析。在单元测试阶段广泛用到插桩技术，即在待测函数内加入功能模块，这些功能模块负责模拟与待测函数调用的函数的行为。单元测试主要用于软件单元中的逻辑错误和功能错误的检测，而不能发现实时性错误和行为错误。

3.1.1.4 集成测试

集成检测是在软件系统进行的测试，可以对软件单位之间接口的准确性进行检测，并检查系统各部位之间是否合理。在应用集成检测方案进行相应检测时，要根据实际检测项目情况为其制订相应的检测计划，然后将相应的单元板模块组合成为子系统或者系统，并且对组合完成的系统进行运行检测，从而判断该系统是否满足其运行需求，

并保障相应各部位的合理性。

3.1.1.5 系统测试

在对软件系统进行开发时，需要对开发的系统进行有效测试，从而使系统与软件应用需求满足相应设计要求。系统测试是对整个系统的测试，将硬件、软件、操作人员看作一个整体，可以发现系统分析和设计中的错误，如安全测试是测试安全措施是否完善，能不能保证系统不受非法侵入。当下我国一些相关部门所应用的检测流程相对复杂，就算流程中只存有非常小的变动，也可能导致检测结果出现一定的偏差。故而，在对系统进行测试前，要对自身实际情况进行全方位考虑，并以此为前提对相应的软件系统程序进行适当调整，从而有效避免系统检测过程中存在问题。而且在对相应系统进行有效检测时，要对软件自身功能以及安全性等各方面进行全方位检测，从而有效保证检测结果的全面性以及客观性。在对相应的系统进行检测时，因为检测流程存有一定的特殊性，为此要根据实际情况为其构建不同的独立检测小组，在对其进行相对有效的检测时，要对系统中存有的组成单元进行实时检测，从而有效保障检测结果准确性。

3.1.1.6 自动化测试

传统软件测试采用人工测试法，在时代的发展下，自动化测试已经得到了广泛应用，大幅提升了测试的效率和准确性。自动化测试的最大特点就是在手工测试基础上，完善多种其他测试工作，如在应用压力测试、大数据测试、崩溃式测试时，如果应用手工测试方式，不仅投入的时间成本较高，也无法确保测试准确性，针对上述情况，都

需要推行自动化测试。软件自动化测试是基于某种程序开展的语言编制测试程序，是在传统手动测试基础上的创新，以电脑自动检测替代传统的手工检测，在测试时，需要将测试流程、自动化编程、测试体系整合。应用软件自动化测试，为程序回归测试提供了便利，如果回归测试程序良好，那么测试结果能够预想，这能够减少测试时间，提升测试效率。另外，软件自动化测试能够重复利用，测试结果、内容高度一致，测试软件能够反复使用。

3.1.1.7 仿真测试

仿真测试，就是模拟软件的真实使用环境，软件配置到真实的使用状态进行的测试，一般发生在产品交付使用前。在智能驾驶开发测试中，仿真测试发挥的作用越来越大。通过仿真建模，模拟器可以方便的模拟测试验证场景，遍历各种极端场景，测试系统在不同环境和场景下的表现，大大降低了测试成本，及时发现问题，提高了系统的可用性。

3.1.2 车用操作系统通用功能测试项

（1）身份鉴别：操作系统应具备用户标识功能、用户鉴别功能、鉴别信息保护功能和用户绑定功能。

（2）访问控制：操作系统应具备访问控制功能和访问控制授权规则。

（3）安全审计：应对与操作系统安全相关的事件生成审计日志，包括运行记录、操作日志、网络流量记录、应用软件运行日志、配置信息等。

（4）数据安全：操作系统应包括数据安全保护机制，包括数据完整性保护、数据保密性保护和剩余信息保护。

（5）存储介质管理：单用户系统中，系统应防止用户进程影响系统的运行；多用户或多系统中，应确保多用户之间采取隔离机制，防止用户数据的非授权访问。

（6）应用软件安全管理：应对第三方应用程序的安装、运行、卸载进行安全规范。

（7）升级功能：操作系统应支持更新升级，至少采取一种安全机制，保证升级过程的安全性；保证升级前后安全属性与升级前一致；升级失败时，系统能够回滚，保证系统完整性。

（8）稳定性：正常工作情况下，车用操作系统应能稳定运行，功耗低、内存占用少，不应造成操作系统死机现象。

（9）兼容性：除自带应用程序，操作系统应提供第三方软件应用接口，支持第三方软件的安装、运行和升级功能，具备无线接入互联网的能力。

3.1.3 车用操作系统差异功能测试项

包含安全车控操作系统、智能驾驶操作系统、车载操作系统、系统软件层、功能软件层等不同的测试特点。安全车控操作系统一般是嵌入式实时操作系统，能利用有限的资源，及时处理多个任务，要求高实时性、高可靠性和安全性。智能驾驶操作系统测试主要关注计算能力、数据吞吐能力、高安全性、高实时性和高可靠性。车载操作系统对安全性和可靠性要求处于中等水平，主要关注易用性、计算能力、

数据吞吐能力、高度的兼容性、灵活性、安全性和可靠性。系统软件是大规模嵌入式系统运行环境，主要关注操作系统内核的实时性和可靠性。功能软件包括感知融合、定位、预测和决策规划各模块，主要关注功能的前置条件，准确性、可靠性、延时、帧率等性能指标，以及插扩式算法冗余和备份机制安全性能标准。

3.1.4 车用操作系统性能测试项

性能指标量化：如启动时间、吞吐量、实时性（响应与延时）、中断响应时间、任务切换时间、资源（占用量、使用率、饱和度）、负载测试、压力测试、稳定性、可靠性。

3.1.5 车用操作系统安全性测试

车用操作系统安全包括功能安全、信息安全、预期功能安全。车用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，所以车用操作系统的功能安全相关系统，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。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。预期功能安全解决系统非故障原因对人造成的危害，ISO26262-1 中提到 ISO26262 对 E/E 系统的标称性能没有要求，对这些系统的功能性性能标准也没有什么要求（例如：主被动安全系统，刹车系统，自适应巡航等），预期功能安全的存在弥补了这部分的遗憾。

3.1.6 车用操作系统测试工具

常用测试工具 CANoe、性能分析工具、仿真测试工具等以及测试系统搭建。

3.2 车控操作系统测试

车控操作系统是应用于智能驾驶功能相关 ECU 的核心软件平台，包含安全车控操作系统、智能驾驶操作系统。根据车控操作系统的不同功能和要求，整体的测试可以分为以下几个部分：

（1）功能测试，包含系统软件和功能软件；

（2）性能测试；

（3）安全测试；

（4）其他测试和认证。

3.2.1 功能测试

3.2.1.1 系统软件层测试

按照系统软件层的总体架构、技术要求与功能要求，系统软件的功能测试主要验证操作系统的功能是否正确完成，是否满足相关的标准和测试规范等，包括不限于：

（1）内核测试：实时操作系统测试；

（2）中间件标准和服务测试，包括功能和标准符合度测试，如 AUTOSAR 兼容性测试，通信接口支持测试；

（3）第三方算法依赖库的支持度测试，包括支持的种类和数目；

（4）面向硬件接口兼容性测试。

3.2.1.2 功能软件层测试

按照功能软件层的总体架构、技术要求与功能要求，功能软件的功能测试主要验证操作系统的功能是否正确完成，是否满足相关的标准等，包括不限于：

（1）传感器抽象接口；

（2）感知融合模块服务接口；

（3）定位模块服务接口；

（4）预测模块服务接口；

（5）决策规划服务接口。

3.2.2 性能测试

车控操作系统的性能测试是测试操作系统在正常环境和系统条件下重复使用是否还能满足性能指标，主要是采用黑盒集成测试的方法来进行，包括不限于：启动时间、吞吐量、实时性（响应与延时）、中断响应时间、任务切换时间、资源（占用量、使用率、饱和度）、负载测试、通信性能、压力测试、稳定性、可靠性等。

3.2.3 安全测试

车控操作系统和传统操作系统的最大不同之处在于对于安全性的要求，由于支撑运行的系统涉及到人的生命财产安全，所以对于其安全性提出了更严格的要求，车控操作系统的安全测试包括含三个方面的内容：

（1）功能安全测试包含可靠性、故障在线监测和处理机制、冗余设计等，测试系统是否能够在系统故障时及时发现和处理，避免或降低故障所导致的风险，参考标准 ISO 26262；

（2）信息安全测试包含身份鉴别、访问控制、数据保护、安全启动、攻击防护等，测试系统是否满足了系统设计安全性目的是否实现了真实性、机密性、完整性、可用性、防抵赖、可授权等需求，参

考标准 ISO 21434；

（3）预期功能安全测试，主要测试在非故障的场景下，系统是否能够消除或者降低因为设计不足或者性能局限导致的安全风险，参考标准 ISO 21448。

3.2.4 其他测试

除了上述针对产品的测试外，对于设计开发还需要按照相应标准要求来实现，并满足相应的等级要求：

- 功能安全：ISO 26262 ASIL 等级；
- 信息安全：ISO 21434 CSMS 认证。

信息技术安全评估准则

除了通用的测试方法，IT 领域还有一项信息安全的通用评估准则《GB/T 18336 信息技术安全评估准则》，定义了通用准则 CC（Common Criteria）。CC 是一种信息技术产品和系统安全性的评估标准，它提供了一组通用的安全功能要求和一组通用的安全保证要求，并在这些保证要求的基础上提供衡量 IT 安全性的尺度（即评估保证级 EAL），使得独立的安全评估结果可以互相比较。

CC 的评估对象（Target of Evaluation, TOE）和范围比较灵活，操作系统是可以应用 CC 评估的软件产品之一。CC 定义了两类的安全要求：

- 功能要求：定义了 IT 产品和系统的安全行为；
- 保证要求：建立对安全功能的信任。

CC 中还有两个重要概念：

- 保护轮廓（Protect Profile, PP）：满足特定用户需求、与一类 TOE 实现无关的一组安全要求（由用户提出要求）；

- 安全目标（Security Target, ST）：作为指定 TOE 的评估基础的一组安全要求和规范（由开发者给出）。

PP、ST 和 TOE 都必须经过评估后才能使用，会给出一个认证评估保证级（Evaluation Assurance Level，EAL），是衡量组件 TOE 保证情况的尺度。它包括七个预先定义的、保证程度依次递增的保证等级。

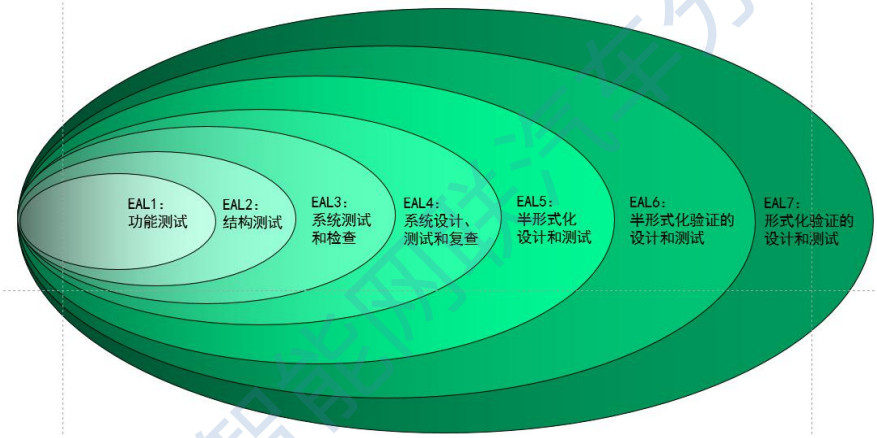

图 7 EAL 评估保证等级

对于代码质量，除了黑盒测试以外，也需要考虑白盒测试的符合度进行评价如判断覆盖和条件覆盖率等。

3.3 车载操作系统测试

3.3.1 功能测试

3.3.1.1 功能软件层测试

与车载操作系统特点相关的功能软件层测试，如互联互通及车内网络测试、多模感知交互、云服务、地图定位、语音交互、媒体播放、升级测试、AI 应用等。

3.3.1.2 系统软件层测试

与车载操作系统特点相关的系统软件层测试，如多系统硬件隔离方案测试、多系统间通信测试、单系统测试、工具链测试等。

3.3.2 性能测试

集成性能测试：如启动时间、吞吐量、实时性（响应与延时）、中断响应时间、任务切换时间、资源（占用量、使用率、饱和度）、信号强度、负载测试、压力测试、稳定性、可靠性等。

3.3.3 安全测试

（一）信息安全测试：如 TEE 可信执行环境、访问控制、安全启动、防火墙、端口扫描、漏洞扫描、安全升级、安全日志、数据加密、安全通信、入侵检测、调试和测试端口保护、协议模糊测试等。

（二）功能安全测试：功能安全是通过增加一些功能措施，确保安全达到可接受的水平。测试的目的是保证系统在各种故障情况下都能够导向一个安全的状态。

功能安全测试技术要求：

（1）软件测试：

1）软件单元测试

a.证明软件单元满足软件单元设计规范；

b.证明软件单元符合软硬件接口的定义；

c.证明软件单元具有一定的鲁棒性；

d.证明软件单元不包含非期望的功能，且满足覆盖度。

2）软件集成测试

a.证明软件实现与软件架构设计的符合性；

b.证明软件接口和软件与硬件接口规范的符合性；

c.证明软件包含了已定义的功能；

d.证明软件具有一定的鲁棒性，具备有效的错误探测以及处理机制；

e.证明软件使用资源与预期消耗资源一致（或小于）。

3）软件安全需求验证

a.验证嵌入式软件符合软件安全需求。

（2）硬件集成测试

a.验证针对硬件安全要求的安全机制实施的完整性和正确性；

b.验证硬件在外部应力下的鲁棒性。

（3）系统测试

1）软-硬件集成测试

a.验证安全机制在软硬件层的正确功能性能、准确性和时序；

b.验证外部和内部接口在软硬件层执行的一致性和正确性；

c.验证对于硬件失效探测机制在软硬件层面上与故障模型有关的诊断覆盖率的有效性；

d.验证安全相关要素在软硬件层的鲁棒性水平。

2）系统集成测试

a.验证功能和技术要求在系统层面的正确执行；

b.验证安全机制在系统层的正确功能性能行为、准确性和时序；

c.外部和内部接口在系统层面执行的一致性和正确性；

d.安全机制在系统层面的失效覆盖率的有效性；

e.验证系统层面的鲁棒性水平。

3）整车集成测试

a.验证功能安全要求在整车层面的正确的执行；

b.验证安全机制在整车层面的正确功能性能、准确性和时序；

c.验证整车层面外部接口实现的一致性和正确性；

d.验证安全机制在整车层面的失效覆盖率的有效性；

e.验证整车层面的鲁棒性水平。

3.3.4 其他测试

可移植性、国际化、标准兼容性、文档资料、认证测试、CTS/VTS等测试项。

3.4 整车集成测试

3.4.1 集成功能测试

集成功能测试是在整车功能维度，将整车功能相关的系统或单元集成在整车环境内进行的测试，主要考察整车功能逻辑在整车集成环境下均能满足设计要求，在功能的各使用场景下均能满足终端用户的预期，以及功能与功能之间交互部分体验的覆盖，同时确保在各个配置下的整车功能均能满足设计要求。由于是在整车环境下，测试的重点与系统集成测试会有不同，整车集成功能测试更贴近终端用户的使用环境。在进行集成功能测试时，要根据实际功能的增减/变更情况为其制订相应的测试范围、测试计划，然后确保相关的系统或单元在整车集成台架或实车环境下的配置，并且进行测试执行，测试过程中

判断该功能是否满足前述的测试目的，如果无法满足需要及时反馈软件开发团队进行问题修复。

3.4.2 集成性能测试

集成性能测试是在整车集成的环境下，对相关整车功能的整车性能表现进行的测试，主要考察相关功能在各使用场景下的整车性能指标和相关整车体验，同时确保在各个配置下的整车性能均能满足设计要求。由于不同的整车功能会有不同的使用条件和用户使用场景，因此需要有不同的判断标准和衡量准则，相应的测试方法也会有不同。在进行集成性能测试时，要根据实际功能的增减/变更情况，以及相关零部件/子系统性能标定的变更情况来制定测试范围、测试计划，然后在整车集成环境（台架、实车）下进行测试执行，测试过程中判断相应的性能是否满足要求，如果无法满足要求则需要及时反馈相应标定和软件开发团队。

3.4.3 仿真台架测试

仿真台架测试和实车测试的区别在于仿真台架测试可以更灵活地实现测试环境的多种配置，更高效的执行各种测试案例，从而达到更好的测试覆盖度。仿真台架测试通过测试需求分析、仿真台架测试环境配置、测试用例设计和测试执行等步骤，实现高覆盖度的测试，为整车集成测试和实际路测做好准备，并根据整车集成测试和实际路测的结果进行回归测试。

在仿真台架中，需对车辆模型、传感器模型、执行器模型、接口模型、测试场景模型等进行配置，以达到尽可能的接近实车的水平，

并通过充分的测试案例的设计和执行，出具测试报告，形成测试结论。

由于仿真台架本身的可配置性，仿真台架测试一般采用自动化测试手段，并可对极限工况进行测试。在测试中，一般操作系统的测试搭载在功能测试上一并进行。

3.4.4 多操作系统协同

多操作系统所在的运行环境实际上是相对独立的，通常采用模块化的开发方式，根据硬件平台的资源需求系统可剪裁、可配置，包含多个独立的功能模块，不同模块可能共享相同的资源。其中，Linux 主要负责上网、刷写、摄像头及语音识别等功能。QNX 主要负责如车载 CAN 通讯、显示及报警提示等功能。Android 主要负责如娱乐、导航、车控及 APP 等功能。而多操作系统之间的交互主要是车载网络信号和流媒体信息的交互。测试方法主要是通过白盒测试验证操作系统内部逻辑结构和处理过程，通过黑盒测试来验证多操作系统之间接口相关的功能需求的有效性，以及通过内核功能测试来验证内存管理、任务管理及中断管理这些项目。

3.4.5 OTA 升级测试

OTA 升级测试主要包括升级前、升级中和升级后测试。

汽车 OTA 升级测试需检测 OTA 功能是否达到升级前期准备的要求，以确保 OTA 升级后续过程能够正常、安全地开展。其中包括车辆状态的检测，识别车辆软件配置、功能状态等，并于后台进行交互，云端下载待升级软件包，并对下载的软件包合法性、完整性进行校验。

汽车 OTA 升级测试需检测 OTA 在升级过程中是否符合技术、安

全、便捷性的要求。升级过程需要有明确的信息提示、安全提醒及注意事项等，并具备足够的安全机制限制特定功能，避免误操作造成车辆财产和人员安全损伤的可能。

汽车升级后测试需要充分保证软件质量满足各方面要求，主要包括软件代码级、单元级、集成级、系统级要求。在整车功能、性能等方面需测试确认达到 OTA 升级预期。

整个 OTA 过程，需要确保各环节安全需求，对通信加密、数字签名、远程攻击防护等方面进行确认，以降低车辆通过 OTA 接口被攻击、伤害的可能。

4 车控操作系统测试体系

4.1 体系架构

车控操作系统作为整个智能网联汽车基础支撑软件平台，服务车辆控制和智能驾驶功能。车控操作系统的测试体系包括对于整体车控操作系统的功能测试、性能测试、安全测试等。同时由于车控操作系统的复杂性，需要在设计开发流程中满足系统的功能安全、信息安全的目标要求，建议的整体测试体系如图 8 所示。

图 8 车控操作系统评测体系

4.2 功能测试

车控操作系统由系统软件和功能软件两部分组成。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 9 所示。

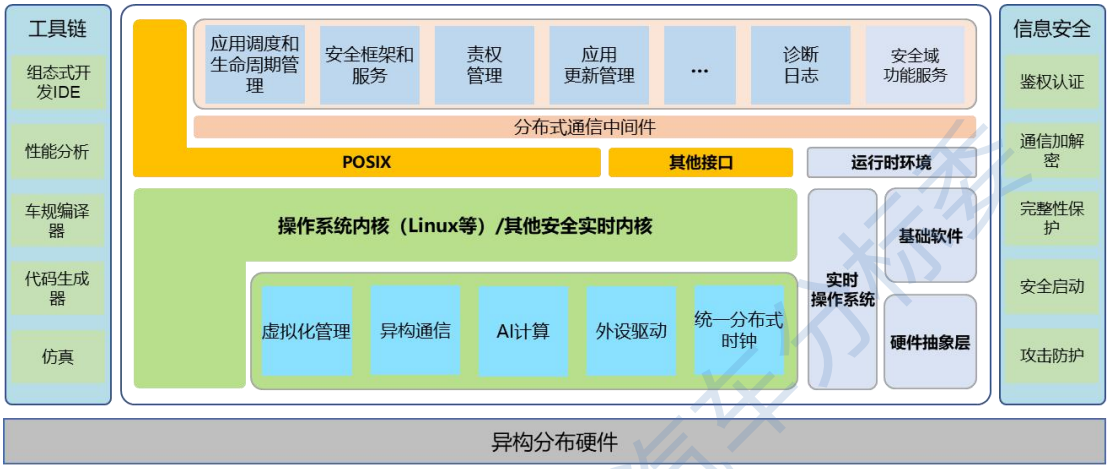

图 9　系统软件架构

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、软件更新服务。

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。

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。车控操作系统功能软件架构如图 10 所示。

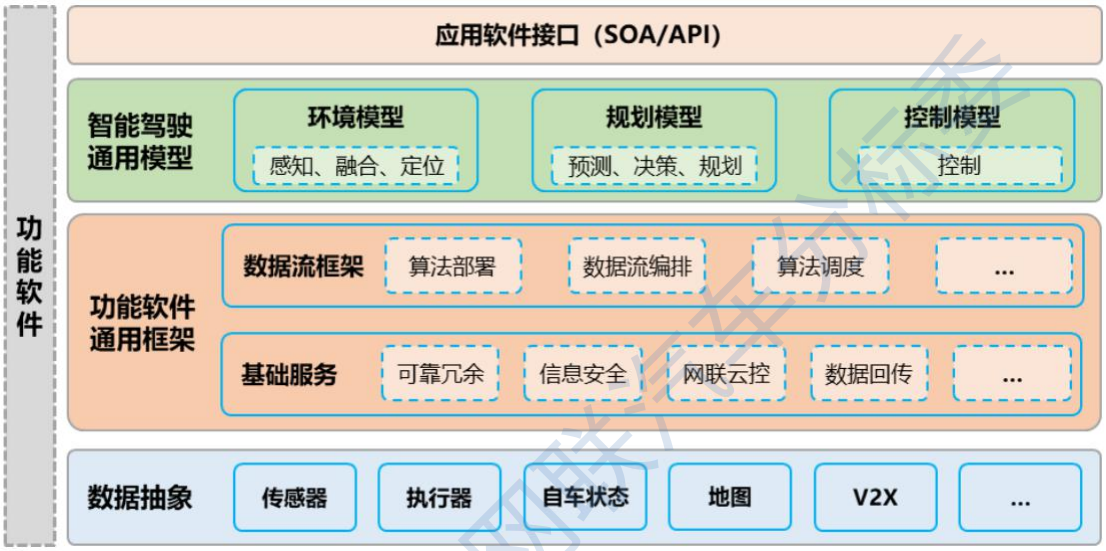

图 10 功能软件架构

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。

（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、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。

（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。

（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。

对于操作系统的功能测试主要包含两部分，一部分是标准符合度的测试，一部分是功能实现验证的测试。

4.2.1 标准符合度测试

标准符合度测试需要参考现相关标准的测试流程和测试用例，验证被测系统的标准符合度。对于智能驾驶和安全车控操作系统都需要满足下列标准并通过相应的测试：

（1）AUTOSAR 兼容性测试，包括 AUTOSAR 经典平台和自适应平台的中间件的标准符合度，支持的标准模块和服务；

（2）通信协议支持测试，测试操作系统支持的通信协议集合，

包括 CAN，以太网，UART 总线，DDS 协议，SOME/IP 协议，LIN，等。

对于智能驾驶系统，还需要满足下列标准：

（1）POSIX 接口兼容性，包括支持的 POSIX 接口的集合，如 PSE 51，PSE52 等；

（2）第三方算法依赖库的支持度测试，包括支持的种类和数目，如 OPENCV，OPENCL 等第三方算法依赖库；

（3）对于其他标准生态的支持，如 ROS 生态等。

对于安全车控系统，还需要满足下列标准：

（1）面向硬件接口兼容性测试，包括对于虚拟化层和硬件抽象模块提供的标准抽象的测试。

4.2.2 安全车控操作系统功能验证测试

对于安全车控操作系统，至少包含以下的功能测试：

（1）OS 的基本功能：任务管理如 Task 的任务调度，资源管理如提供资源互斥访问机制等，事件管理如提供事件同步机制，定时机制驱动任务调度，MCU 的中断管理，调度管理如提供任务的时间编排，各种保护机制包括提供时间保护，内存保护，服务保护和信任接口保护，错误保护等；

（2）OS 的运行时环境：执行调度提供软件和基础软件的实体调度服务；通信接口即基础软件的通信功能支持，提供多种通信机制；

（3）OS 的基础软件服务：系统服务包含模式管理，故障管理等服务；存储服务包含存储服务、存储硬件抽象服务；通信服务包括

Com 通信，SOMEIP，DDS 面向服务的通信协议；诊断服务；时间同步包括时间同步管理和通信总线的时间同步协议支持。

4.2.3 智能驾驶操作系统功能验证测试

对于智能驾驶操作系统，至少包含以下的功能测试：

（1）OS 的基础功能测试：

1） 内存管理功能测试，包含对于内存分配，内存回收，内存访问机制，内存保护机制等相关内存管理功能的验证；

2） 调度管理功能测试，通过模拟不同的突发事件，验证系统对于应用和进程的调度策略，对各个中断、进程和线程的操作功能；

3） 设备管理功能测试，包括设备中断处理及驱动框架的测试；

4） 文件管理功能测试，验证系统的内存和文件管理，包括文件系统的创建，删除，更新等管理操作，包括文件大小，文件命名规则等的测试和验证；

5） 网络管理功能测试，验证车内网络变化，通过网联对设备的唤醒和休眠等功能；

6） 调试功能测试，包含对 GDB 等调试工具的支持；

7） 基础功能库测试，包括 C/C++的基础库支持测试。

（2）应用框架基础功能测试：

1） 执行管理即应用/服务生命周期管理功能测试，验证操作系统对于进程和应用组件的创建，激活，暂停，删除等管理功能；

2） 通信中间件功能测试，通过测试服务器和测试终端，模拟数据的收发功能，验证采用不同的通信协议，包括不限于 DDS，

SOME/IP，进程间通信等，服务器和客户端正确接受和发送；

3） 日志管理功能测试，通过模拟不同的事件，来测试整个系统对于日志季度的完整性，准确性和及时性等；

4） 诊断管理测试，测试上位机和诊断服务的通信测试对于标准的诊断服务的支持情况；

5） 升级功能测试，验证被测系统具备 OTA 差分升级和整包升级功能，测试系统作为 OTA 的服务端，验证升级服务的正常、失败、部分失败等场景下的处理情况，及时上报升级进度和结果。

对于智能驾驶操作系统的功能软件的测试包括以下服务接口：

1）传感器抽象接口通过规范和模块化各类自动驾驶传感器，通过共性提取形成对外连接的标准服务接口，为感知融合和定位提供基础，智能传感器包含毫米波雷达，超声波雷达，激光雷达和摄像头，覆盖传感器的目标，特征和检测级别的输出，以及传感器的健康状态和性能状态；

2）感知融合接口是感知融合模块提供的服务接口，感知融合服务完成了对传感器抽象模块的输入数据进行融合，完成对物理世界的数字呈现。具体的服务接口包括目标识别和跟踪、道路结构、交通标志、静态目标、可行驶空间、自车状态、泊车位信息和环境模型等，和传感器抽象的目标接口不同，感知融合接口是融合多种传感器的输出；

3）定位服务接口是定位模块提供的服务接口，根据高精度地图、传感器等信息输入提供自车位置，服务接口包括自车定位和地图等；

4）预测服务接口是预测模块的输出，依据环境信息和交通参与者历史测量信息，对其他交通参与者的未来行驶意图和轨迹进行预测，服务接口包括行为预测和轨迹预测等；

5）决策规划接口是决策规划模块的输出，根据感知融合、自车定位和交通参与者预测等信息输入来完成自车行驶轨迹的决策和规划，并根据决策结果输出车辆控制命令。服务接口包括导航信息、行为决策、轨迹规划和运动控制等；

6）执行器抽象接口和传感器抽象接口类似，规范和模块化各类车辆执行器，通过共性提取形成对外连接的标准服务接口。其可通过标准服务接口接收自动驾驶系统决策规划信息，并结合车辆当前的行驶状态来完成车辆实际控制，并反馈最终执行结果。

4.3 性能测试

性能测试是通过自动化的测试工具模拟多种正常、峰值以及异常负载条件来对系统的各项性能指标进行测试。性能测试可以体现系统的能力，可以考虑一定的评分机制来对不同的系统进行比较，从而帮助用户更好的了解整个系统，为用户的选择提供一个参考。

4.3.1 可靠性测试

根据车控操作系统的实时性、确定性等特点，需要利用负载测试和压力测试的方法来对车载情况的调度性能、通信性能和可靠性进行测试。通过负载测试，确定在各种工作负载下系统的性能，目标是测试当负载逐渐增加时，系统各项性能指标的变化情况。压力测试是通过确定一个系统的瓶颈或者不能接受的性能点，来获得系统能提供的

最大服务级别的测试。如可以采用 72 小时不间断的压力测试记录整个系统的运行情况。

4.3.2 安全车控操作系统性能测试

对于智能驾驶操作系统的性能测试可以通过以下的几个方面来进行性能测试：

（1） 时间特性的测试

1） 系统启动时间测试，即系统冷启动到基础服务运行的启动时间；

2） 通信时延和抖动测试，通过采用一发一收、多发多收测试在不同条件下不同通信技术如 CAN 以太等通信的时延和时延抖动；

3） 时间同步信息，包括不同的通信技术的时间同步精度性能和支持时间同步协议等。

（2） 任务调度性能测试

1） 任务切换时间，即测试系统在两个独立的、处于就绪态并且具有相同优先级的任务之间切换所需要的时间；

2） 中断响应时间，即从接收到中断信号到操作系统做出响应，并完成进入中断服务进程所需要的时间。

（3） 资源占用测试

即测试 OS 对于系统资源的开销状况，包括处理器资源和存储资源占用情况等；

（4） 网络性能测试

即测试系统的网络吞吐率，在不同负载情况下网络传输处理能

力。

4.3.3 智能驾驶操作系统性能测试

对于智能驾驶操作系统的性能测试可以通过以下的几个方面来进行性能测试：

（1） 时间特性的测试

1） 系统启动时间测试，即系统冷启动到基础服务运行的启动时间；

2） 通信时延和抖动测试，通过采用一发一收、多发多收测试在不同条件下不同通信技术如 CAN 以太网等通信的时延和时延抖动；

3） 时间同步信息，包括不同的通信技术的时间同步精度性能和支持时间同步协议等。

（2） I/O 性能测试

即对于存储 I/O 的性能进行测试，测试文件顺序和重复读写的性能；

（3） 资源占用测试

即测试 OS 对于系统资源的开销状况，包括处理器资源和存储资源占用情况等；

（4） 网络性能测试

即测试系统的网络吞吐率，在不同负载情况下网络传输处理能力。

4.4 安全测试

安全包括功能安全和信息安全，是车控操作系统和其他操作系统的主要区别之一。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，预期功能安全 ISO 21448 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。未来再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，测试覆盖率和条件覆盖率需要满足相应的 ASIL 等级要求；

（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）预期功能安全

车控操作系统需要支撑从辅助驾驶到高级别自动驾驶的应用功

能运行，也提供应用相应的功能软件组件和服务，预期功能安全的需求和测试也是需要考虑的。操作系统的相关部分的开发设计需要遵循ISO 21448 的开发流程和设计要求，考虑在系统受限和合理可预期的误用情况下操作系统是否可以将风险控制在可接受的范围内，保障车辆的运行安全，因为相应的标准还在研制过程中，需要在后续的研究讨论中考虑后续对于预期功能安全的测试。

4.4.1 功能安全测试

车控操作系统作为智能网联汽车软件功能安全的核心和基础，在开发设计当中也需要满足相应的功能安全要求：

（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。

智能驾驶操作系统和安全车控操作系统服务应用场景对于功能安全的要求也是有区别的，根据智能驾驶系统和车辆控制系统的安全需求分别进行测试。

对于智能驾驶操作系统，需要测试的功能安全特性至少包含：

（1） 健康管理功能测试，测试对操作性系统的正常运转情况的监控；

（2） 端到端通信安全测试，测试端到端通信完整性保护；

（3） 分区隔离功能测试，测试对于不同功能安全需求的应用之间的安全域分区隔离；

（4） 安全启动功能测试，测试系统启动时 Safety 进程启动和运行情况。

对于安全车控操作系统，需要测试的功能安全特性至少包含：

（1） 看门狗监控，提供整个系统的运行监控机制；

（2） 端到端通信安全测试，测试端到端通信完整性保护；

（3） 应用隔离机制，提供测试操作系统对于不同安全需求和安全域的隔离。

4.4.2 信息安全测试

车控操作系统是车控控制相关的硬件资源的直接管理者，所有应用软件都是基于车控操作系统来运行的，车控操作系统的信息安全是整个智能网联汽车信息安全的基础。车控操作系统信息安全测试验证是实现安全操作系统的一个极为重要的环节，只有经过安全评测和验

证，才能确定操作系统的可信度，进而确定整车的安全性。

车控操作系统的信息安全测试验证可以分为开发设计阶段要求和产品测试。车控操作系统的安全是一个不断演进的过程，通过采用威胁分析与风险评估（TARA）方法对于操作系统的信息安全威胁进行分类，设定处理机制和方法，同时在产品上市后也需要有完整的信息安全漏洞管理机制和应急响应机制，迅速完善产品，避免利用操作系统的安全漏洞进行攻击。安全测试的目的就是检查安全机制是否完整的实现了安全策略，如果功能实现不符合设计的安全策略，则可以判断系统存在安全漏洞，需要进行相应的升级。对于安全测试横跨单元测试，集成测试和系统测试。

对于车控操作系统产品的信息安全测试可以延续现有通用的产品的测试方法，至少包含形式化验证、代码功能块检查、Fuzz 测试、渗透测试等方法。

（1）Fuzz 测试，即将自动或半自动生成的随机数据输入到一个程序中，并监视程序异常，如崩溃，断言（assertion）失败，以发现可能的程序错误，比如内存泄漏。模糊测试常常用于检测软件或计算机系统的安全漏洞。

（2）渗透攻防，一种通过模拟使用黑客的技术和方法，挖掘目标系统的安全漏洞，取得系统的控制权，访问系统的机密数据，并发现可能影响业务持续运作安全隐患的一种安全测试和评估方式。渗透测试以实战的方式来检验系统的防御体系和应急响应计划的有效性。渗透测试需要考虑系统的漏洞，应用代码类的漏洞，配置不当类的漏

洞，数据安全保护漏洞，业务逻辑缺陷等。

对于智能驾驶操作系统，信息安全测试至少包含以下机制和特性的测试：

（1）访问控制机制，即按照最小化权限原则，对于应用范围的权限进行控制；

（2）身份认证，即对服务调用者的身份进行认证，保证访问者的身份不会被仿冒；

（3）地址随机化，即针对缓冲区溢出的安全保护，通过对堆、栈、共享库映射等线性区布局的随机化，通过增加攻击者预测目的地址的难度，防止攻击者直接定位攻击代码位置，达到阻止溢出攻击的目的；

（4）流量控制，即通过可变的滑动窗口的随机化，对不同应用和服务的通信流量进行管控；

（5）安全标记，即系统出现问题，可以回滚到之前的一个安全节点；

（6）安全审计，即系统需要通过对于安全标准的符合性测试；

（7）可信路径，即对于系统启动路径/应用程序访问系统服务的可信要求；

（8）完整性监控，即系统完整性被破坏后，提供系统运行情况的记录；

（9）通信加解密，通过对通信过程的加解密，保护通信内容。

对于安全车控操作系统，信息安全测试至少包含以下机制和特性

的测试：

（1）安全机制包括通信的加解密服务，基于安全上下文（SecOC）的安全通信服务；

（2）通信的机密性，如 TLS 加密的支持等。

4.5 其他测试和认证

由于车辆的安全性涉及到用户的生命财产安全，所以传统汽车的开发设计需要满足功能安全要求。在软件定义汽车的趋势下，车控操作系统作为智能网联汽车软件平台的核心和基础，其开发设计也需要满足功能安全要求。在新的网联形式下，车辆的信息安全也越来越重要，因信息安全原因最终也会导致对车辆功能安全的影响，从而影响用户的生命财产安全。所以对于智能网联汽车的车控操作系统，包括智能驾驶和安全车控操作系统，需要通过相应的信息安全评测，比较成熟的有功能安全 ASIL 等级认证和信息安全 EAL 认证，例如：

（1）对于智能驾驶操作系统，核心模块需通过 ASIL-D 认证和 CC EAL 5+认证；

（2）对于安全车控操作系统，核心模块需通过 ASIL-D 认证和 CC EAL 5+认证。

5 车载操作系统测试体系

5.1 体系架构

车载操作系统是用户和车载硬件的接口，同时也是车载硬件和上层软件的接口。车载操作系统的功能包括管理车载系统的硬件、软件及数据资源，控制程序运行，改善人机界面，为上层软件提供支持，

让车机系统的资源，以及接收到数据、信号、音频、视频最大限度地发挥作用，提供各种形式的用户界面，使驾驶员有一个好的驾驶环境。车载操作系统的测试体系包括对于操作系统的功能测试、性能测试、安全测试、接口测试等，整体测试体系如图 11 所示。

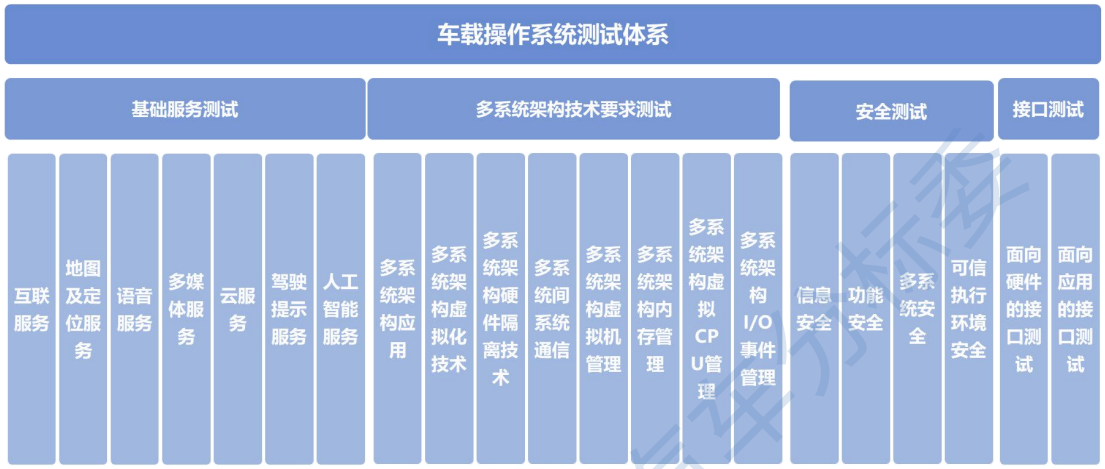

图 11　车载操作系统测试体系

5.2 基础服务测试

5.2.1　互联服务

车载操作系统应支持与 T-Box、OBD、Tuner、V2X-OBD 等多车载终端互联服务。车载操作系统应支持与个人移动端、云端的信息交互互通。系统应支持通过无线（如 Wi-Fi、蓝牙、NFC）或有线（如 USB）等方式，与个人移动端投屏互联，实现娱乐智能服务、辅助控制服务等；系统应内置常用 MQTT、CoAP、HTTPS 等网络协议，具备与云端互通能力，连接各种云服务。车载操作系统互联服务主要关注系统与终端的连接、通讯互通、信息同步等测试。

（1）系统与终端的连接，包括不同的设备、终端（如 Carplay、Carlife、Hicar、Android Auto 等手机有线及无线的互联、WiFi、BT、

USB 等），连接的响应性能及压力测试。不同的终端、不同的连接方式连接的响应时间不一样，响应时间越快给用户的体验越好；连接的压力测试指标可以从汽车的使用寿命及用户的使用场景去考虑。

（2）系统与终端的通讯互通，如与云端的数据传输交换能力、与设备的通讯、不同协议的切换。

（3）信息同步体现在系统与设备端数据传输与显示的时效。如个人移动端与系统的投屏互联，移动端交互到系统的同步显示，时延越小越好。

5.2.2 地图及定位服务

车载操作系统应支持地图服务（位置服务），为包括地图在内的各类应用程序提供位置相关的 API。地图服务模块可结合 GNSS、各类传感器以及车辆信号数据，采用复杂的数据融合及滤波算法，实现准确的位置信息求解，为包括地图在内的各类应用程序提供位置相关的 API。云端可实时获取上传的位置数据，部署各类服务，将位置信息与第三方服务或数据挖掘的数据进行融合，为用户提供所需的与位置相关的增值和个性化的服务。地图及定位服务的测试指标主要包括定位精度、搜星性能。

5.2.3 语音服务

车载操作系统应支持语音服务，其架构可分为语音形象、语音编程框架、语音服务模块、语音开放平台和语音自学习系统五部分。

（1）语音形象：语音形象是一个特殊的系统内置应用，提供了所有交互展示型的功能，并向上层的语音类应用提供 API。比如：当

有语音输入（ASR）操作时，语音形象应用会将识别出来的文字显示在屏幕上。当有语音播报（TTS）时，语音形象应用会处于播报态。语音形象应用通过接收下层语音框架上报的多种状态信息（唤醒态、识别态、处理态、播报态等），展示相应状态所对应的可视化界面。

（2）语音编程框架：提供语音 API 支持基于 CAF 编写的语音应用（CloudApp）或车载小程序语音应用。

（3）语音服务模块：语音框架还提供了多个语音服务，主要是提供本地语音语义能力和与云端能力对接，主要包括：声学前端和语音唤醒模块（WUW），语音识别模块（ASR），语音合成模块（TTS），语音声纹模块（VPR），语义理解模块（NLU），对话管理模块（DM），语义生成模块（NLG）等。整个架构是一套端云一体组件化可插拔的技术架构。

（4）语音开放平台：支持第三方应用开发者自助式编写离线/在线的 NLU、DM、NLG 等技能，通过技能管理还可进行可视化编辑、管理等工作。

（5）语音自学习系统：通过将语音唤醒、语音识别等用户交互的音频数据进行定期的回流，并对获取的大量语音数据进行分析和训练，可以得到更好的相关模型参数，通过闭环优化系统，定期将这些模型参数及相关资源进行下发，使最终用户可以感觉到语音定期会更新很多新的能力，语音唤醒、语音识别、语义理解等效果也变得更好，而且能提供很多个性化的优化效果，用户体验也会变得更好。

5.2.4 多媒体服务

车载操作系统应支持基础的图像、音频、视频的编解码播放和控制服务。音频、视频的编解码能力体现在不同格式的媒体的兼容，播放和控制测试指标包括播放质量（噪音、帧率、音质、流畅度、清晰度、音视频同步等）、播放资源占用（网络带宽、CPU 占用）。

5.2.5 云服务

（1）自动升级服务（OTA）

车载操作系统应支持自动升级服务（OTA），通过网络自动检测到新版本（如新功能、安全补丁等）并下载安装，为新功能、安全补丁的发布提供升级通道。

自动升级服务架构如图 12 所示，由客户端与服务端两部分构成。客户端以系统级服务的方式运行在操作系统中，支持服务端检测系统版本、下载升级包、安装升级包等操作。服务端应包括如下模块：

1）操作控制台：主要为配置管理员提供发布版本、上传升级包的操作功能；

2）版本查询服务：向客户端提供版本检测查询服务，要能稳定支撑大量设备的并发查询请求；

3）升级包下载服务：向客户端提供升级包下载服务，通常需要依赖分布式的文件存储并接入数据中心以提高下载速度；

4）数据存储：结构化的版本数据存储以及非结构化的升级包文件存储；

5）其他：升级过程全链路数据分析模块、版本升级效果统计模

块等，以帮助掌握升级过程并跟踪新版本升级情况和效果。

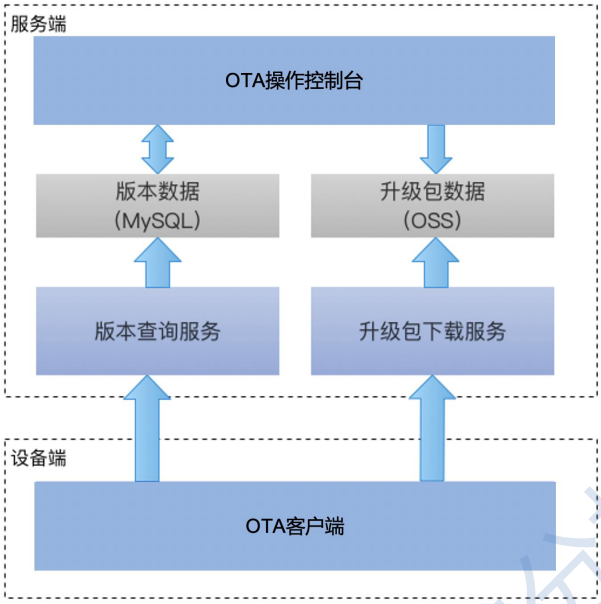

图 12 自动升级服务（OTA）架构

自动升级测试关注升级包的上传、下载、升级校验及更新过程。升级包的上传、下载速度反映了升级时间的长短，一般升级时间越短，用户体验更好及用户操作出错的概率也较低。升级过程升级校验的准确性、更新过程的成功率都是自动升级服务需要关注的测试点。

（2）账号

账号包括系统账号、系统中安装的应用账号和服务账号。不同的账号服务可有各自的独立的用户界面、操作流程以及鉴权方式。

系统账号是面向互联网汽车的车载操作系统中云+端资源的串接器，通过账号统一接入服务端的服务资源和计算资源，实现数据共享和服务融合，以及多移动端的信息流转和互通。系统账号应提供分层次的应用授权访问机制，通过服务端鉴权保证服务端访问的安全性和合法性。系统账号应支持对本地资源的管理，通过服务端鉴权保证在本地增加、删除以及切换账号对应本地用户资源并访问该资源的安全

性和合法性。

车载操作系统应支持的账号管理机制包括：

1）统一管理和开放账号服务

2）匹配系统账号和本地用户资源管理

3）支持多系统账号

4）账号管理与账号服务解耦

5）多账号数据隔离

账号管理的测试有账号的登录及切换、账号数据的保存、多账号数据隔离、账号安全、账号权限管理等。

1）账号切换：不同的账号登录与注销；多账号同时登录；账号数量最大临界点处理。

2）账号数据保存：不同账号数据的保存；账号数据与用户资源匹配;账号数据的更新与时效性。

3）多账号数据隔离：不同账号的数据不受影响和冲突。

4）账号安全：登录安全（验证码、登录超时、一个账号多系统登录、错误登录限制）、数据安全。

5）账号权限管理：不同账号级别区分，不同级别的账号权限设置与管理。

5.2.6 驾驶提示服务

车载操作系统可支持简单的驾驶提示服务，如 360 环视、倒车影像、盲区提示、车窗控制等。结合 CAN 消息的硬件抽象、标定、电源策略、时间策略和亮度策略，基于图像拼接算法等技术，完成单一

车控功能或组合式车控功能。驾驶提示服务关注的是响应时间、准确性。测试项包括 360 环视、倒车影像的响应时间，激活进入的时间越短越好；以及摄像头图像的显示、拼接还原度。

5.2.7 人工智能服务

随着车机芯片或者异构 AI 芯片的算力增强，智能座舱的智能化程度越来越高，对于视觉感知、语音交互、网联通信等功能需求的不断提升，车载操作系统的人工智能服务能力也许相应提升。

车载操作系统需支持结合 TensorFlow、MXNet、Caffe、PyTorch 等多种机器学习框架，同时再利用量化推理、自动计算调度优化、异构编译执行等技术，来支撑本地 AI 功能开发。基于以上技术可设计语音识别、自然语言处理、图像识别、文本分类、地图导航、搜索引擎等 AI 模块引擎，支持 AR 导航、盲区检测、驾驶员疲劳检测、FaceID、手势识别、语音图像理解等人机交互 AI 应用，满足这些应用的各类性能指标要求。

AI 服务关注数据模型在对待正常数据、边界数据、异常数据作为输入时，模型的输出是否能够符合期望。模型在处理数据时的效率（学习过程，CPU 占用率、内存消耗等）。自优化测试：将测试集分成两部分（或多部分），首先输入第一部分，观察结果，然后再输入第二部分，然后再次输入第一部分，观察输出是否有优化的体现。

5.3 多系统测试

5.3.1 多系统架构应用

一体化座舱平台是目前的主流趋势，用统一的处理器芯片支持车

内的智能交互设备，包括中控信息系统、液晶仪表、HUD 等，实现一芯多屏多系统，座舱域集中整合、降低硬件复杂度和成本。一芯多屏多系统主要有以下优点：一，将前装多个屏幕上的芯片功能整合在一块高性能芯片上以降低成本和功耗；二，使用一块计算芯片为各个组件的互联互通打下了一定基础；三，使用一块计算芯片能简化项目开发复杂度和降低供应商整合难度；四，提供统一芯片组对芯片厂家而言意味着更高的产业链附加值和话语权。

智能座舱的统一芯片控制使得计算资源可以集约化利用，同时根据功能安全 ISO 26262 标准要求,不同关键数据、代码与功能也应满足不同安全等级要求（ASIL），虚拟化和硬件隔离两种多系统架构方案可兼顾实现这两方面的要求。

5.3.2 多系统架构虚拟化技术

虚拟化技术实现硬件资源（CPU、内存、I/O、存储）的虚拟化共享和分区隔离，允许多个 guest os 共享硬件平台，并且保证安全隔离。支持不同类型 guest os 运行（如 rtos、andriod、linux），系统资源分配动态负载均衡，保证满足仪表、中控等多屏应用的要求。

常用虚拟化技术 Hypervisor，运行在基础物理服务器和操作系统之间的中间软件层,可允许多个操作系统和应用共享硬件，也可称为 VMM（Virtual Machine Monitor），即虚拟机监视器。采用虚拟化和硬件辅助虚拟化技术路线，支持硬件分区、负载隔离、实时处理、安全引导和运行时完整性，支持整个软件栈的安全隔离，支持 guest os 之间通信。关键技术如下：车载操作系统多系统所需的技术能力和特

点包括：

（1）硬件系统资源分配和共享（静态和动态）；

（2）外设资源的分配和共享（静态和动态）；

（3）域隔离，保证操作系统之间的独立性；

（4）支持多操作系统间通信，支持系统间的数据共享；

（5）支持系统安全和功能安全相关方案；

（6）管理各操作系统优先级，实现自适应分配最大化利用硬件资源，并满足对时间敏感需求和系统服务质量的要求；

（7）管理各操作系统生命周期，包括启动，运行，关闭，重启等状态；

（8）Hypervisor 的存在应保证其实时性，不应对各操作系统性能产生影响；

（9）具有不同安全性和安全级别的应用程序可以在相同的硬件上运行，通过硬件或软件分区相互保护；

（10）全局电池管理。

基于 Hypervisor 模式特征可以划分为 2 类：

（1）全虚拟化

全虚拟化也称为硬件虚拟化或者 Type-1 型虚拟化技术，通过提供系统虚拟机（例如模拟类似于真实硬件的整个系统），允许未经修改的客户机操作系统作为客户机运行，该模式需要借助硬件的虚拟化支持，例如 X86 架构 AMD-V/Intel VT，ARMv8 和 Power 架构的虚拟化 profile 等。其优势在于客户操作系统不需要进行任何修改就可以使

客户操作系统正常运行，并且它们并不知道自己在虚拟化环境下运行。这种虚拟化技术使用简单，具有很好的兼容性，但这种虚拟化方式由于需要捕捉客户操作系统发出的敏感特权指令，进而通过虚拟机管理器来模拟系统的特权指令，这个过程将降低指令的执行速度。

（2）半虚拟化

由于全虚拟化技术对硬件性能的要求较高，并且运行效率不高。因此提出了半虚拟化技术解决方案。在半虚拟化方案（Type-2 型虚拟化技术）中，虚拟机系统（客户 OS）的内核需要经过特殊修改，把特权指令改成对虚拟化层 API 的调用。在全虚拟化的基础上，把客户操作系统进行了修改，增加了一个专门的 API，这个 API 可以将客户操作系统发出的指令进行最优化，即不需要 Hypervisor 耗费一定的资源进行翻译操作，因此 Hypervisor 的工作负担变得非常的小，因此整体的性能也有很大的提高。半虚拟化技术可以减少模拟执行特权指令造成的性能开销，另外这种技术还可以优化 I/O 访问和操作，运行速度基本可达到本机速度。

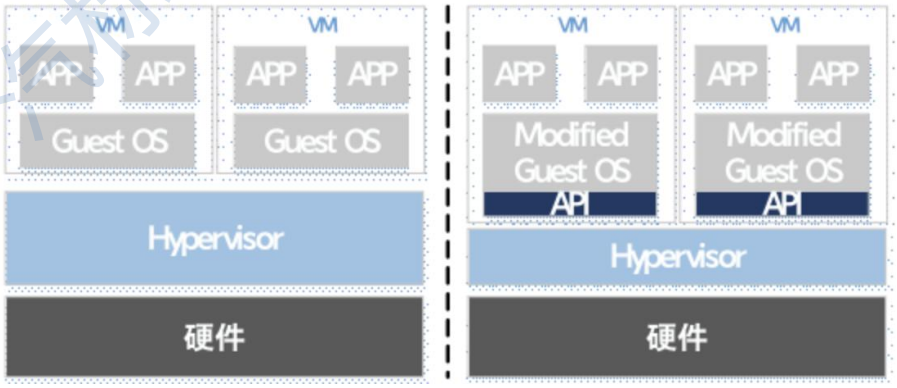

图 13 全虚拟化和半虚拟化技术架构对比

5.3.3 多系统架构硬件隔离技术

硬件资源通过硬件分区的方式进行划分和管理，硬件资源的所属分区拥有对该资源的访问和管理权限，其他分区不能对该资源进行操作。针对具备内存保护单元的多核 SOC，通过创建两个系统的硬件分区，将不同的硬件资源划分到两个分区，并将不同 CPU 内核划分到两个分区，可实现两个分区上的 CPU 分别独立运行各自操作系统。

SCFW 固件是 SCU（System Control Unit）中运行的软件程序，SCU 为硬件基本功能提供抽象。SCFW 固件负责管理硬件资源的所有权和访问权限。SCFW 的资源管理功能通过创建分区的形式划分 SOC 资源，并控制主事务属性和外围设备访问权限。SCFW 支持的功能有：

（1）管理系统资源，如 SOC 外围设备（Resource），内存区域（Memory）和引脚等。

（2）允许将资源划分为与不同执行环境关联的不同所有权组。

（3）允许所有者配置对资源的访问权限。

（4）提供硬件强制隔离。

5.3.4 多系统间系统通信

（1）基于具备硬件隔离方案的控制单元提供的内部通讯

（2）基于 Hypervisor 提供的通讯机制

共享内存方式是 Hypervisor 域间通信效率较高的机制。由 Hypervisor 定义一段物理内存，允许多个操作系统将该物理内存连接到各自的地址空间中，系统中特定的进程可以访问共享内存中的地

址。如果某个操作系统向共享内存内写入数据，所做的改动将立即影响到可以访问同一段共享内存的其它操作系统。关键技术如下：

1）虚拟化异步事件机制；

2）基于共享内存的域间批量数据传输机制；

3）虚拟化域间共享内存资源管理；

4）审查数据传输的合法性；

5）把共享内存区域从输出进程的页表中摘除；

6）把共享内存区域从输出虚拟机的全局内存空间表中摘除，此时共享内存区域归 VMM 所有；

7）把共享内存区域加入到输入虚拟机的全局内存空间表中；

8）把共享内存区域加入到输入进程的页表中。

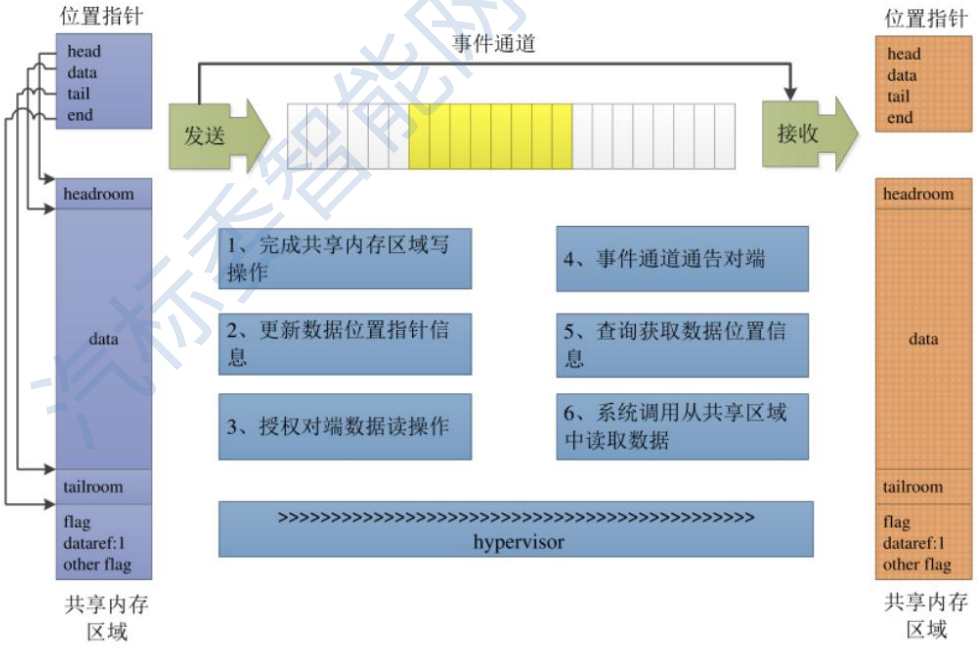

图 14　基于 Hypervisor 提供的共享内存通讯机制

虚拟网络方式是由 Hypervisor 建立起虚拟的内部网络，并为每个操作系统分配唯一的地址和标识符，各操作系统可以基于 socket 及各

类网络通讯协议实现系统间通讯，如新型 IPC 机制 FDBUS。FDBUS 开发框架适用于在定制系统上开发交互复杂的分布式项目，包括车载以太网连接域开发、Hypervisors 上多个 Guest OS 之间通信开发。FDBUS 具有如下特点：

1）分布式：基于 TCP socket 和 Unix Domain socket（UDS），既可用于本地 IPC，也支持网络主机之间的 IPC；

2）跨平台：支持 Window、Linux 和 QNX；

3）高性能：点对点直接通信，不通过中央 Hub 或 Broker 转发；

4）安全性：能够为 server 的方法调用也事件广播配置不同等级的访问权限，只有权限足够高的 client 才能特点方法和接收特定事件；

5）服务名解析：server 地址用名字标识，通过 name server 注册服务和解析名字，从而 server 可以在全网络任意部署；

6）支持跨平台的中间件开发框架，包括组件：Thread 模型、Event Loop、基于 Job-Worker 的线程间通信、基于 Event Loop 的 Timer、基于 Event Loop 的 watch、Mutex、Semaphore、Socket、Notification；

7）IPC 采用 Client-Server 模式，支持通信模式：带超时的同步请求-答复、带超时的异步请求-答复、无答复的命令请求、注册-发布模式，实现多点广播；

8）IPC 消息采用 Protocol buffer 序列化和反序列化，支持 IDL 代码生成，高效简便；也支持 raw data 格式，便于大量数据传输；

9）可靠的心跳和重连机制，确保无论网络状况如何，无论哪个服务重新上线或重启，通信各方都能保持连接。

另外，L4 系列微内核中，也可基于灵活页域间通信。灵活页代表 L4 微内核的对象资源抽象，包括物理内存页面、外设页面等。采用灵活页作为域间数据传输载体，通过授权、映射将数据传输给接收方，中间不再需要任何数据拷贝，性能开销非常低。

5.3.5 多系统架构虚拟机管理

在多系统架构中，客户机操作系统运行在虚拟机中，系统需提供虚拟机管理功能以便对客户机操作系统进行管理。

5.3.6 多系统架构内存管理

为保证客户机操作系统能够在虚拟机中正常运行，需要分配一定的内存给虚拟机供客户机操作系统使用，所以多系统架构需提供内存管理功能：

（1） 内存的申请和释放；

（2） Cache 策略设定；

（3） Cache 相关操作。

5.3.7 多系统架构虚拟 CPU 管理

CPU 是操作系统的基础计算资源，多系统架构中硬件的 CPU 资源通常会被抽象成虚拟 CPU，虚拟 CPU 会被分配给虚拟机使用，功能包括：

（1） 虚拟 CPU 的创建和销毁；

（2） 虚拟 CPU 的恢复运行；

（3） 虚拟 CPU 的中断触发；

（4） 虚拟 CPU 的状态读写。

5.3.8 多系统架构 I/O 事件管理

在多系统架构中，部分硬件设备需要供多个客户机操作系统共享使用，这需要提供一种机制利用有限的硬件设备实现虚拟设备，共享给多个虚拟机使用，所以多系统架构需提供 I/O 事件管理功能：

（1） 虚拟机 Trap 地址设置；

（2） 虚拟机 Trap 事件接收。

5.4 安全测试

5.4.1 信息安全

鉴于车载操作系统内核功能的复杂性，车载操作系统应构建完整的信息安全防护机制，降低内核漏洞对系统安全的危险，强化内核的安全防护能力。系统应具备多域隔离、一致性检查、安全启动、安全存储、安全输入、加密文件系统等技术，应具备系统镜像完整性及合法性验证机制、系统镜像防回退校验功能。

（1）内核完整性保护

可分为静态完整性保护和运行时完整性保护，确保内核不被篡改或在篡改后能够被迅速发现。测试流程：1）获取操作系统安全启动信任根存储区域的访问方法和地址，使用软件调试工具写入数据，重复多次检查是否可将数据写入该存储区域；2）提取操作系统签名，使用软件调试工具对签名进行篡改，将修改后签名写入到车载终端内的指定区域，检查是否正常工作。评价指标：支持 kernel 模块加载时的签名验证，防止内核模块被替换。

（2）数据隐私及加密

根据车载FOTA特点提供密钥方案和文件级加密方案，提高不同量产车辆产品的安全性，提升本机对data分区不同文件的加密保护。测试流程：1）以非授权身份通过车载端访问存储在车内系统的敏感数据；2）以非授权身份尝试对存储在车内系统的敏感数据进行修改。评价指标：对存储在车内系统的用户敏感数据（私人信息/身份，用户账号，支付账号，位置信息等）设置安全加密及权限访问控制，防止未授权的访问、篡改、删除和检索。

（3）安全隔离机制

提供空间隔离机制，将内核与外部的驱动模块进行安全隔离，阻断攻击者从物理存储中获取安全容器内的文件内容。测试流程：1）检查是否使用可信执行环境解决方案，是否对关键资源（例如：密钥、证书）的访问设置访问控制；2）对关键资源（例如：密钥、证书）的进行非授权访问和修改。评价指标：支持可信执行环境（TEE），为基于敏感数据的关键应用提供安全执行空间，控制对关键资源的访问，保护资源和数据的保密性和完整性，对抗非授权访问和篡改等多种攻击。

（4）安全启动机制

源于硬件可信根的校验链，逐级校验下一阶段代码和文件系统的完整性，防止系统运行的代码被恶意篡改。测试流程：1）检查安全技术防护说明文档，车载端系统启动时是否采用安全机制，在验证操作系统签名并判定通过后，再从可信存储区域加载车载端操作系统；2）检测是否可以通过更改设置或者修改代码旁路安全启动；3）采取

多种方式破坏信任链，检测安全启动是否仍然执行。（非法签名，无签名，启动程序篡改等）。评价指标：系统启动时使用可信机制，验证操作系统签名并判定通过后，从可信存储区域加载车载操作系统，避免加载被篡改或签名被破坏的操作系统。

（5）安全存储机制

密钥管理模块与硬件安全能力结合，进一步提升文件和密钥存储的安全性。测试流程：1）检查是否能获取本地存储的密钥、访问密码、安全日志等敏感信息；2）如果存在以上信息，检查是否能篡改本地存储的密钥、访问密码、安全日志等敏感信息。评价指标：密钥应存储在 TEE 可信环境中，防止未授权的访问、篡改、删除和检索。

5.4.2 功能安全

车载操作系统从应用角度分为中控系统、仪表系统和 T-box 系统等，应确保车载操作系统满足 ISO 26262 安全标准，不同应用车载操作系统的功能安全等级要求不同，如仪表需满足 ASIL-B 的安全等级要求。

针对仪表等功能安全等级要求较高的系统软件，为防止系统内部多层级、多模块的单点软件失效等软件故障或瞬时硬件故障造成严重后果，可对系统软件进行冗余设计，来确保系统功能的稳定可靠。应对潜在故障的方法较多，下面列出了一些最常用的技术：

（1）多样化检查器：使用另一条电路检查主电路是否发生故障。举个例子，检查器可以为中断控制器计数，持续记录人为及系统引起的中断总数。

（2）完整锁步复制：该技术主要用于特殊的处理器，对一个 IP 元件（如一个处理器）进行多次实例化，利用循环产生操作延迟，生成时间和空间冗余。大容量存储通常由多个实例共享，以降低所需面积。尽管这一技术非常可靠，但也极为昂贵。

（3）选择性硬件冗余：这个方案里，只有硬件的关键部分可以复制，如仲裁器。

（4）软件冗余：因为硬件冗余通常非常复杂，而且会产生间接成本，是对资源的不合理使用。硬件运算的替代方法就是，在多个处理器内核上运行同一次计算，检查结果是否匹配。

（5）错误检测和校正码是另一种为人熟知的技术，通常被用于保护存储器和总线。代码类型多种多样，但目标只有一个，既通过少量附加位获得更高冗余，无需复制所有底层数据。汽车系统中，这一尖端技术可以利用足够多的冗余检测出一个存储字的 2 位错误；并支持错误修正。

在系统软件冗余设计中，关键任务处理时，可使用两种不同的冗余通道软件算法来执行相同的任务，比较两种冗余通道的输出结果，如发现差异，则生成故障消息。再者，也可在时间维度上以冗余的方式执行安全关键任务，使用同一软件算法（限定相同执行条件）在一定时间内执行多次同样任务，比较多次执行结果，如出现差异，则执行相关的纠正操作，消除瞬时故障。功能安全主要测试指标如下：

（1）故障处理时间间隔

控制器需要及时处理故障，因此需要评估故障发生时的最大处理

时间。根据 ISO 26262，只评估一个故障或两个故障同时发生。故障处理时间间隔（FHTI）包括故障检测时间间隔（FDTI）和故障反应时间间隔（FRIT），如下图所示：

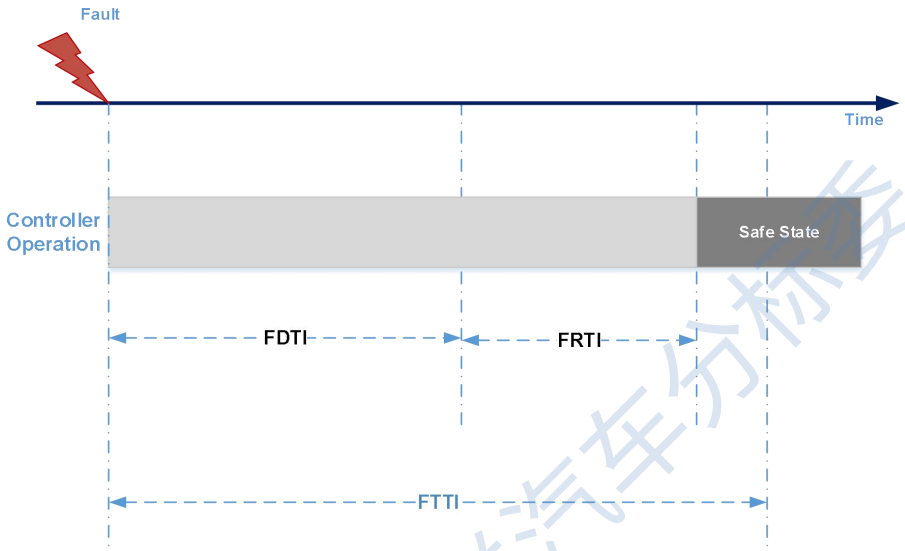

图 15　故障处理时间

如果在一段时间内发生两次故障并满足以下条件将需要最大 FHTI：

- 这两个故障属于级联故障；
- 最远的故障先出现；
- 确认最远故障后，立即发生第二故障。

下图显示了发生的两个级联故障，Fault1 是最远的故障，Fault2 是第二个故障。

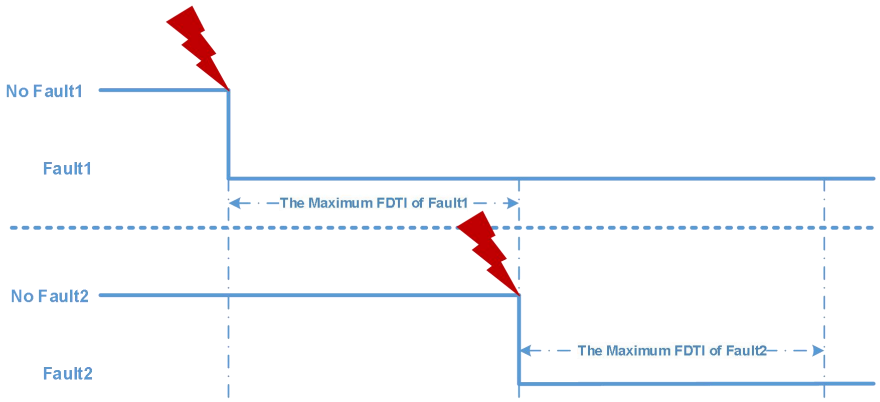

图 16 两个级联故障发生序列

（2）安全状态策略

当系统检测到故障时系统如何处理以确保系统安全。由于不同的故障对系统的影响是不同的，因此需要定义系统安全状态、系统模式状态、功能工作状态以及这些状态之间的关系。

测试流程：1）通过软件模拟制造故障，或通过硬件制造故障；2）验证系统是否出发相应故障；3）确认故障处理时间间隔。

5.4.3 多系统安全

多系统安全应满足如下要求：

1）多系统安全启动的整体流程应具备信任根和完整的信任链，确保多系统代码和数据的完整性和可靠性；

2）多系统域隔离安全应确保多操作系统间的空间隔离（如内存、Cache 等）、时间隔离（如硬实时、时间窗配置等）和外设的隔离（如 TrustZone、IOMMU、SMMU、设备半虚拟化等技术）；

3）多系统权限模型应确保虚拟机的每项资源都能够通过独立的权限进行管理，系统中无具备最高等级权限的虚拟机；

4）应具备多系统间通信访问控制的能力（例如数据包监测、连接管理等），在检测到攻击行为时有能力断开与攻击源的通信连接；

5）多系统驱动框架对驱动提供有限防护（例如基于内核安全机制检测、前后端驱动隔离等技术）和彻底的防护（例如基于微内核技术将不同驱动彻底隔离）。

5.4.4 可信执行环境安全

5.4.4.1 TEE 硬件层安全

（1）密码算法

测试目的：验证 TEE 硬件层所执行的密码算法符合相应的算法标准（如 3DES、RSA 等）。

测试过程：检查 TEE 硬件层的对称密码或非对称密码算法是否符合相应的密码算法标准。

通过标准：TEE 硬件层应按照相应的算法标准正确执行密码算法。

（2）随机数生成器

测试目的：验证 TEE 硬件层随机数生成器产生的随机数具备足够的随机性。

测试流程：验证 TEE 硬件层产生随机数的质量，进行 GB/T 32915、AIS 20/AIS 31、NIST SP800-22 或 FIPS PUB 140-2 标准化的测试。

通过标准：满足标准化测试的要求。

（3）异常检测机制

测试目的：验证 TEE 硬件层具备足够的异常检测机制，且能按照其容错机制进行相应的处理。

测试流程：检查 TEE 硬件层产生异常时是否具备相应的检测机制，包括环境异常检测、程序执行异常检测和逻辑模块异常检测，并按照 TEE 硬件层的容错机制进行相应的处理。

通过标准：TEE 硬件层能够对常见的异常进行检测，并按照其容错机制进行相应的处理。

（4）存储器访问控制

测试目的：验证 TEE 硬件层存储器访问控制策略的有效性。

测试流程：检查 TEE 硬件层存储器访问控制策略。

通过标准：TEE 硬件层具备且遵循存储器访问控制策略。

（5）总线传输安全

测试目的：验证 TEE 硬件层具备足够的保护机制以防止其传输系统的物理位置被探测或修改。

测试流程：尝试定位 TEE 硬件层传输系统，并评估其物理位置探测的难易等级和总线加密防护强度。

通过标准：TEE 硬件层应具有较强的物理防护机制防止传输系统位置被探测，传输系统应采用总线 加密、极性反转、总线校验等安全机制，防止总线信息泄露。

（6）寄存器管理

测试目的：验证 TEE 硬件层寄存器管理策略的有效性。

测试流程：检查 TEE 硬件层寄存器管理策略。

通过标准：TEE 硬件层具备且遵循寄存器管理策略。

（7）启动流程

测试目的：验证 TEE 硬件层启动过程中，对其工作环境、完整性等安全机制具备有效的自检机制。

测试流程：评估 TEE 硬件层的启动过程。

通过标准：TEE 硬件层启动过程中，能够对其工作环境和完整性执行有效检验。如果发生异常，TEE 硬件层应处于复位或不可用状态。

（8）调试

测试目的：验证 TEE 硬件层具备安全调试的能力。

测试流程：检查 TEE 硬件层的调试接口的相应控制策略。

通过标准：TEE 硬件层应防止通过调试接口非法访问内部的敏感信息。

（9）中断处理

测试目的：验证 TEE 硬件层具备中断处理的能力。

测试流程：检查 TEE 硬件层的中断级别与分类的管理机制。

通过标准：TEE 硬件层应保证在特定的运行状态或异常报警等情况发生时，按照优先级进行中断处理。

（10）物理防护

测试目的：验证 TEE 硬件层具备足够的物理防护机制。

测试流程：检查 TEE 硬件层中敏感信号的物理防护机制和布线约束规则。

通过标准：TEE 硬件层中敏感信号线应采用底层金属层布线或金

属层遮挡等物理防护机制。

（11）侧信道分析

测试目的：验证 TEE 硬件层在执行密码算法时具备抗侧信道分析的能力。

测试流程：参考 TEE 硬件层设计文档，尝试对其进行侧信道攻击。

通过标准：TEE 硬件层在执行密码算法时无法通过侧信道分析的方式获取密钥等敏感信息。

（12）错误注入

测试目的：验证 TEE 硬件层具备防错误注入攻击的能力。

测试流程：参考 TEE 硬件层设计文档，尝试对其进行错误注入攻击。

通过标准：TEE 硬件层应具备防错误攻击的能力，无法通过错误注入的手段获取密钥等敏感信息。

（13）可信时钟源

可信时钟源是可信执行环境系统中使用的看门狗计时器和可信执行环境调度计时器的硬件基础。可信时钟源应在设备加电启动后立即运行，且不能被禁用、关闭、篡改等，避免因外部干扰等原因导致可信执行环境系统内的编程状态被破坏。可信时钟源的测试评价方法如下：

1）审查厂商提交的文档，检查可信执行环境的可信时钟源设计；

2）检查设备启动过程中对可信时钟源的配置；

3）在设备运行时，尝试禁用、关闭、篡改可信时钟源。

（14）可信调试单元

可信调试单元负责整个设备的调试功能，其硬件基础应满足以下要求：

1）可信调试单元硬件子系统应保证所有侵入式调试机制能够被禁用；

2）可信调试单元硬件子系统应保证所有调试机制（包括非侵入式和侵入式）可以被设定为只有可信执行环境才能够使用；

3）可信调试硬件单元子系统应保证所有调试机制（包括非侵入式和侵入式）可以被富执行环境使用；当可信调试硬件单元子系统被设定为富执行环境和可信执行环境都能够访问时，调试子系统不允许访问或修改可信执行环境中的任何寄存器与存储器；

4）可信调试硬件单元子系统的寄存器使用过程中应保证持续供电，或者能够保证寄存器在系统断电前被完整的保存并在系统恢复供电时完整恢复。

可信调试单元的测试评价方法如下：

1）审查厂商提交的文档，检查可信执行环境的可信调试单元设计；

2）尝试通过侵入式调试机制使用可信调试单元；

3）设置可信调试单元的所有调试机制（包括非侵入式和侵入式）仅被可信执行环境使用，尝试在可信执行环境之外使用调试机制；

4）设置可信调试单元的所有调试机制（包括非侵入式和侵入式）

可被富执行环境和可信执行环境使用，尝试通过调试机制访问或修改可信执行环境中的寄存器与存储器；

5）设置可信调试单元硬件子系统的寄存器后，尝试执行系统断电，上电后，重新读取相应寄存器，并与断电前的寄存器值进行比较。

5.4.4.2 TEE 系统软件层安全

（1）应用识别

测试目的：验证 TEE 系统软件层具备 TA 和 CA 识别的能力。

测试流程：参考 TEE 系统软件层设计文档，执行应用识别操作。

通过标准：TEE 系统软件层应确保 TA 的 ID 的唯一性，并能够区分 TA 和 CA。

（2）密钥管理

测试目的：验证 TEE 系统软件层具备密钥管理机制。

测试流程：参考 TEE 系统软件层设计文档，执行密码管理中相关操作。

通过标准：TEE 系统软件层应根据密钥创建者的设定，限定密钥的用途。TEE 系统软件层应根据符合相关标准的特定密钥生成算法和特定密钥长度来产生密钥，并根据符合相关标准的特定密钥分发方法来 分发密钥。TEE 系统软件层销毁密钥后，已销毁的密钥信息不可被获得。

（3）TEE 标识

测试目的：验证 TEE 系统软件层具有唯一 ID，且不可被更改。

测试流程：尝试获取或使用、对比、更改 TEE 系统软件层的 ID。

通过标准：TEE 系统软件层应具有唯一 ID，且该 ID 是不可被更改。

（4）启动流程

测试目的：验证 TEE 系统软件层具备安全启动的能力。

测试流程：执行 TEE 系统软件层的启动流程，并尝试对其进行攻击。

通过标准：TEE 系统软件层应通过一个安全的初始化流程启动。

（5）实例化时间单向性

测试目的：验证 TEE 系统软件层具备实例化时间不可逆的能力。

测试流程：多次获取 TEE 系统软件层的实例化时间，并将结果进行比较。

通过标准：TEE 系统软件层应提供 TA 的实例化时间，该时间在 TA 实例化期间是不可逆的。

（6）安全操作

测试目的：验证 TEE 系统软件层具备正确执行安全功能，且不受异常状态影响的能力。

测试流程：执行 TEE 系统软件层的安全功能，确认安全功能的执行不会受到异常状态的影响。

通过标准：TEE 系统软件层应实现安全功能的正确操作，不受异常情况的影响，具备安全服务的访问控制，不会泄露敏感信息。

（7）随机数

测试目的：验证 TEE 系统软件层使用的随机数具备足够的随机

性。

测试流程：验证 TEE 系统软件层产生随机数的质量，进行 GB/T 32915、AIS 20/AIS 31、NIST SP800-22 或 FIPS PUB 140-2 标准化的测试。

通过标准：TEE 系统软件层应使用 TEE 硬件层随机数生成器产生的随机数，且该随机数满足标准化测试的要求。

（8）运行时数据机密性

测试目的：验证 TEE 系统软件层具备保证运行时数据机密性的能力。

测试流程：检查 TEE 系统软件层运行时的数据，验证其能够防止非法泄露。

通过标准：TEE 系统软件层应保证运行时数据的机密性，TA 数据和密钥未发生非法泄露。

（9）运行时数据完整性

测试目的：验证 TEE 系统软件层具备保证运行时数据完整性的能力。

测试流程：检查 TEE 系统软件层运行时的数据，验证其能够防止非法篡改。

通过标准：TEE 系统软件层应保证其固件和运行时数据、TA 的代码和数据等在易失性存储器中运行时未被非法篡改。

（10）TA 真实性保护

测试目的:验证 TEE 系统软件层具备验证 TA 代码真实性的能力。

测试流程：检查 TEE 系统软件层对 TA 代码真实性的校验过程，验证该校验过程的有效性。

通过标准：当 TA 被装载到安全存储区域后，TEE 系统软件层应执行真实性验证，验证通过 TA 后方可被加载。

（11）TA 隔离

测试目的：验证 TEE 系统软件层具备隔离 TA 的能力。

测试流程：执行 TA 操作，尝试访问其他 TA 的资源，验证 TA 之间隔离机制的有效性。

通过标准：TEE 系统软件层应保证每个 TA 仅能访问自己的执行和存储空间。

（12）TEE 数据保护

测试目的：验证 TEE 系统软件层具备保证 TEE 持久性数据的真实性、完整性、机密性的能力。

测试流程：尝试获取、篡改 TEE 持久性数据。

通过标准：TEE 系统软件层应确保 TEE 持久性数据的真实性、完整性、机密性。

（13）TEE 隔离

测试目的：验证 TEE 系统软件层具备保护其执行和存储的空间和资源的能力。

测试流程：尝试通过 REE 和 TA 获取 TEE 系统软件层执行和存储的空间和资源。

通过标准：TEE 系统软件层应防止 REE 和 TA 获取其执行和存

储的空间和资源。

（14）可信存储

测试目的：验证TEE系统软件层具备为TA提供可信存储的能力。

测试流程：尝试执行可信存储操作，并获取、篡改TA存储的数据和密钥。

通过标准：TEE系统软件层应为TA的通用数据/密钥提供可信存储服务，保证其机密性、真实性和一致性，对存储内容的修改操作具备原子性。TEE系统软件层的可信存储应与设备绑定。

（15）回滚保护

测试目的：验证TEE系统软件层具备防止未经授权回滚的能力。

测试流程：执行TEE系统软件层的回滚操作。

通过标准：只有通过认证的用户方可执行TEE系统软件层的回滚操作。

（16）TA持久时间的单向性

测试目的：验证TEE系统软件层具备为TA提供持续性时间的能力。

测试流程：获取TEE系统软件层为TA提供的持续性时间。

通过标准：TEE系统软件层应能够为TA提供持续性时间，且不受TEE系统软件层复位的影响。

（17）调试

测试目的：验证TEE系统软件层的调试功能只有通过认证的用户才能使用。

测试流程：尝试使用 TEE 系统软件层的调试功能。

通过标准：只有通过认证的用户方可使用 TEE 系统软件层的调试功能。

（18）可信根

可信根为可信执行环境建立及运行提供支撑，可以是硬件、代码和数据。可信根应具备以下安全要求：

1）应具备机密性、完整性、真实性三个基本安全特性，能够为可信执行环境系统的安全鉴证、安全度量和安全存储提供支持；

2）应提供访问控制机制，保证未经授权的用户不能访问和篡改可信根的数据和代码。

可信根的测试方法如下：

1）审查厂商提交的文档，检查可信执行环境的可信根设计；

2）检查可信执行环境系统的安全鉴证、安全度量和安全存储过程，验证可信根是否提供了机密性、完整性、真实性等安全特性；

3）尝试使用未经授权的用户访问和篡改可信根的数据和代码，验证访问控制机制是否有效。

（19）可信虚拟化系统

可信虚拟化系统应具备以下能力：

1）创建、删除等动态管理可信执行环境内虚拟机的能力；

2）管理可信执行环境中虚拟机内部 CPU、内存、中断、外设等硬件资源的能力；

3）可信执行环境内虚拟机之间应具备互相通信和数据交换的能

力。

可信虚拟化系统应具备如下安全要求：

1）应保证可信执行环境内各虚拟机仅根据其所分配的权限访问相应的资源，不能越权访问；

2）应保证可信执行环境系统自身及内部虚拟机加载和运行过程的正确性与完整性；

3）应保证可信执行环境系统自身及内部虚拟机数据与代码的真实性和完整性；

4）可信执行环境内虚拟机之间的通信应具备一定的访问控制策略。可信虚拟化系统应避免赋予内部虚拟机最高等级权限，单一虚拟机出现崩溃或安全隐患时，不应影响到可信执行环境系统自身及内部其他虚拟机的正常工作。

可信虚拟化系统的测试方法如下：

1）审查厂商提交的文档，检查可信虚拟化系统的设计；

2）在可信执行环境内动态创建和删除虚拟机，并尝试管理虚拟机内部 CPU，内存，中断，外设等硬件资源；

3）在可信执行环境内创建多个虚拟机，验证虚拟机之间是否支持互相通信及进行数据交换，检查虚拟机之间通信的访问控制策略，根据策略内容分别访问策略所允许访问和不允许访问的虚拟机，验证策略是否有效；

4）在可信执行环境内创建多个虚拟机，尝试越权访问其他虚拟机的资源；

5）检查可信执行环境内部虚拟机的加载和运行过程，尝试篡改相应数据和代码；

6）检查可信虚拟化系统内部虚拟机的等级权限设置，验证是否赋予虚拟机最高等级权限。

（20）可信操作系统

可信操作系统应具备常规操作系统中的进程管理、内存管理、设备管理、文件管理等基本系统功能。可信操作系统要求在访问控制、身份鉴别、数据完整性、可信路径等方面满足相应的安全技术要求。包括：

1）应保证可信应用及可信服务仅根据其所分配的权限访问相应的资源，不能越权访问；

2）应保证系统自身、可信服务与应用启动的正确性与完整性；

3）应保证系统自身、可信服务与应用数据和代码的真实性和完整性；

4）应具备可信应用之间、可信应用与可信服务之间的访问控制能力；

5）对于系统权限的管理，应避免赋予可信服务与应用最高权限，避免单一可信应用与服务出现异常时，影响系统内核及其他可信应用与服务的正常工作。

可信操作系统的测试方法如下：

1）审查厂商提交的文档，检查可信操作系统的设计；

2）检查可信操作系统的访问控制策略，尝试使用可信应用及可

信服务分别访问策略所允许和不允许访问的资源，验证策略是否有效；

3）检查可信操作系统自身、可信服务与应用启动过程，尝试篡改启动代码和绕过完整性校验过程；

4）检查可信操作系统自身、可信服务与应用数据和代码的真实性和完整性保护机制，尝试篡改相应数据和代码；

5）检查可信应用之间、可信应用与可信服务之间的访问控制策略，尝试使用可信应用及可信服务分别访问策略所允许和不允许访问的可信应用和可信服务，验证策略是否有效；

6）检查可信操作系统内部可信服务与可信应用的权限设置，验证是否赋予可信服务与可信应用最高权限。

5.4.4.3 TA 安全

（1）安全审计

测试目的：验证 TA 提供方法记录安全相关事件的方法，以便帮助管理者发现潜在的攻击或发现由于 TA 安全特性的错误配置而陷入易被攻击的状态。

测试流程：审查厂商提交的文档，验证厂商已声明 TA 具备安全审计功能，并执行安全审计功能。

通过标准：TA 具备安全审计功能，能够为相关可审计事件生成审计记录。

（2）启动流程

测试目的：验证 TA 的启动流程具备自检的功能。

测试流程：审查厂商提交的文档，验证厂商已声明 TA 具备自检功能，并启动 TA。

通过标准：TA 的启动流程应具备自检功能，包括敏感信息和密钥的检查。

（3）内存清除

测试目的：验证 TA 具备内存清除的功能。

测试流程：审查厂商提交的文档，验证厂商已声明 TA 具备内存清除功能，并使用 TA 处理敏感信息。

通过标准：TA 在使用完敏感信息后应立即对其进行清除。

（4）逻辑异常

测试目的：验证 TA 具有抵抗逻辑操纵和修改的结构，保护自己免受逻辑异常侵害。

测试流程：审查厂商提交的文档，验证厂商已声明 TA 具备逻辑异常的处理，对 TA 进行逻辑异常攻击。

通过标准：TA 可以抵抗通过逻辑操作对安全特性的攻击，不会受到逻辑异常的影响而泄露任何敏感信息。

（5）密钥使用

测试目的：验证 TA 具备安全使用密钥的能力。

测试流程：审查厂商提交的文档，验证厂商已声明 TA 对密钥的使用满足安全要求，并执行 TA 的密钥操作。

通过标准：TA 对密钥的使用遵循 TEE 系统软件层的安全要求，或者根据密钥的用途使用。

（6）生命周期

测试目的：验证 TA 具备检测当前生命周期状态和限制命令执行的能力。

测试流程：审查厂商提交的文档，验证厂商已声明 TA 具备生命周期管理功能，并在 TA 的不同生命周 期执行命令。

通过标准：TA 的设计和执行应保证命令只用于与之对应的生命周期中。

（7）安全操作

测试目的：验证 TA 具备正确执行安全功能，且不受异常状态影响的能力。

测试流程：审查厂商提交的文档，验证厂商已声明 TA 能够正确执行安全功能，不受异常状态影响， 并在异常状态下执行 TA 安全功能。

通过标准：TA 能够遵循 TEE 系统软件层的安全要求正确执行安全功能，或者执行安全功能时能够不 受异常状态影响。

（8）可信存储

测试目的：验证 TA 具备安全存储数据和密钥的能力。

测试流程：审查厂商提交的文档，验证厂商已声明 TA 具备可信存储功能，并执行 TA 的可信存储。

通过标准：TA 能够遵循 TEE 系统软件层的安全存储要求，或者能够对其存储的数据和密钥的机密性、真实性和一致性提供保护，同时对存储内容的修改操作具备原子性。TA 的可信存储应与设备绑定，

即数据只能由创建时同一设备、TEE 上的经授权的同一 TA 进行访问和修改。

（9）回滚保护

测试目的：验证 TA 具备防止未经授权回滚的能力。

测试流程：审查厂商提交的文档，验证厂商已声明 TA 具备回滚保护能力，并执行 TA 的回滚操作。

通过标准：只有通过认证的用户方可执行 TA 的回滚操作。

（10）防重放

测试目的：验证 TA 具备防止重放攻击的能力。

测试流程：审查厂商提交的文档，验证厂商已声明 TA 具备防重放保护，并尝试进行重放攻击。

通过标准：TA 能够保护其资源并防止重放攻击，当受到重放攻击时，能够作出相应的响应，其资产不会受到危害。

5.4.4.4 TEE 外部设备安全

可信外设指对驱动控制与数据采集有一定安全或隐私要求的一类外设。在使用可信外设前，系统必须被切换到可信执行环境内，以阻止任何停留在富执行环境的恶意代码监控和记录数据的输入。对于采集与处理过程不能够完全由可信执行环境控制的外设，宜采用以下两种方式进行处理：

1）加密传输方式，即在可信外设和可信执行环境之间建立加密通道，保证数据在传输过程中的机密性、完整性、真实性；

2）监控方式，通过可信执行环境对富执行环境及整个设备的安

全状态进行严格检测和监测，及时控制风险。

可信外设的测试方法如下：

1）审查厂商提交的文档，检查可信执行环境的可信外设设计；

2）在可信执行环境使用可信外设时，尝试通过富执行环境监控和记录可信外设的交互数据；

3）对于采集与处理过程不能够完全由可信执行环境控制的外设，检查可信外设和可信执行环境之间的数据传输保护机制，验证可信执行环境是否具备对富执行环境及整个设备的安全状态的监测和响应机制。

（1）指纹模块-传感器

测试目的：验证指纹模块-传感器具备唯一识别 ID，且不可被篡改。

测试流程：检查传感器是否具备唯一识别 ID，该 ID 应体现型号及版本等信息，验证其无法被篡改。

通过标准：每颗传感器应具有唯一 ID，表示其型号及版本信息，且不可被篡改。

（2）指纹模块-硬件模块

1）合法性认证

测试目的：验证指纹模块-硬件模块具备合法性认证的能力。

测试流程：检查硬件模块与上游设备的合法性认证过程，验证该过程校验硬件模块的合法性，并且 硬件模块具备唯一 ID。

通过标准：指纹模块-硬件模块应有唯一 ID，并且与其上游设备

之间进行合法性认证。

2）指纹模板存储

测试目的：验证指纹模块-硬件模块或其上位机驱动对指纹模板的存储提供保护。

测试流程：检查指纹模板的存储过程，验证指纹模板是加密存储的，只有需要调用模板时才解密使用。

通过标准：指纹模板应加密储存，需要调用模板时解密使用。

3）数据传输安全

测试目的：验证指纹模块-硬件模块与上游设备之间的数据传输安全性。

测试流程：检查硬件模块与上游设备之间的数据传输协议，验证该协议能保证通信数据的机密性、完整性和真实性，能防止重放攻击，并且使用的安全协议版本不低于 TLS 1.2。

通过标准：应保证通信数据的机密性、完整性和真实性，并能防止重放攻击，使用的安全协议版本 应不低于 TLS 1.2。

4）密钥管理

测试目的：验证指纹模块-硬件模块具备密钥管理能力。

测试流程：检查硬件模块的密钥体系，验证该体系的合理性以及用于加密指纹信息和进行设备合法性认证的密钥的有效性。

通过标准：指纹模块-硬件模块应建立合理的密钥体系，用于加密指纹信息和进行设备合法性认证。

5）密码算法

测试目的：验证指纹模块-硬件模块的密码算法满足相关算法标准（如 3DES、RSA 等）。

测试流程：检查对称密码或非对称密码算法是否符合相应的密码算法标准，验证使用的密钥长度符合安全要求。

通过标准：指纹模块-硬件模块应提供对应的对称和非对称算法功能，密钥长度应符合安全要求。

6）随机数

测试目的：验证指纹模块-硬件模块产生的随机数具备足够的随机性。

测试流程：验证指纹模块-硬件模块产生随机数的质量，进行 GB/T 32915、AIS 20/AIS 31、NIST SP800-22 或 FIPS PUB 140-2 标准化的测试。

通过标准：指纹模块-硬件模块产生的随机数满足标准化测试的要求。

7）固件更新

测试目的：验证指纹模块-硬件模块具备固件更新的能力。

测试流程：检查硬件模块的固件更新过程，验证该过程能够加密验证更新固件的完整性，并且在验证失败时，能够拒绝固件更新。

通过标准：指纹模块-硬件模块应加密验证更新固件的完整性，如果验证未通过，应拒绝固件更新。

8）逻辑异常

测试目的：验证指纹模块-硬件模块具备处理逻辑异常的能力。

测试流程：检查硬件模块的逻辑异常处理过程，验证其在处理逻辑异常时不会泄露敏感信息。

通过标准：指纹模块-硬件模块应不受逻辑异常的影响，包括但不限于非预期命令序列、未知命令、错误设备模式下的命令、错误参数或数据等。

9）内存清除

测试目的：验证指纹模块-硬件模块具备内存清除的能力。

测试流程：检查硬件模块的敏感信息处理过程，验证敏感信息使用完成后会立即清空内部保存的敏感信息。

通过标准：指纹模块-硬件模块的敏感信息在使用完成后，应及时清空其在内部保存的敏感信息。

5.5 接口测试

传统车载 ECU 模块的硬件平台和操作系统基本自成一体，各个模块的软件接口和系统基本不能兼容，替换不同厂商的模块和系统往往就意味着局部架构的改变。今后的操作系统应能够提供统一的接口，为今后产品的封装和模块化提供有力的支持。

5.5.1 面向硬件的接口测试

面向硬件的接口主要有面向 SoC 的硬件接口、面向整车信号的接口，面向外围 IC 的接口以及面向 IoT 设备的互联接口等。面向硬件的接口将操作系统面向车规级芯片的接口标准化，使得操作系统在调用硬件资源时，不需要了解硬件的内部逻辑及资源分配情况，只需要按照标准的接口进行参数配置即可实现对硬件的调用，从而实现对

于不同硬件的适配，保证系统较高移植性。其中，多核异构架构适配性技术要求包括异构平台硬件体系的融合驱动和异构平台并行计算框架。

测试流程：1）审查厂商提交的文档，检查资源抽象的设计；2）检查异构硬件资源抽象是否支持对各种设备类型分别定义数据、控制和事件接口，包括摄像头硬件资源抽象、通用的车控管理系统和接口硬件资源抽象等，尝试使用车载管理系统获取资源，尝试使用硬件接口资源抽象获取数据；3）检查整车信号资源抽象是否支持对整车电子设备进行接口硬件抽象，提供 MCU 的 CAN 消息上行/下行接口，尝试采用 CAN 消息进行硬件访问；4）检查外围 IC 的资源抽象是否支持对蓝牙等外围 IC 的硬件资源抽象和接口，尝试采用蓝牙设备建立连接等。

5.5.2 面向应用的接口测试

面向应用程序的接口主要有 UI 控件、图形图像处理、多媒体、网络、蓝牙、摄像头、音频、机器学习、位置、车身信号、本地数据库、应用管理、系统等。面向应用程序的接口使得应用程序在调用操作系统的基础服务时，不需要了解其内部逻辑，只需要按照统一的接口进行参数配置即可实现应用程序与操作系统的交互，从而进一步实现应用程序与操作系统的解耦，促进应用生态发展。

测试流程：1）在操作系统的基础上，编写应用程序，根据接口文档说明调用各个接口，给接口传入一定的参数，使接口执行预期的功能；2）判断接口功能执行的结果是否符合预期，检查接口返回值

是否正确，并检查接口调用过程中是否引发系统或者程序异常，以及资源使用率是否在指定的范围内。

评价指标：接口覆盖率，100%覆盖关注的接口。

6 车用操作系统测试案例

6.1 安全车控操作系统测试

6.1.1 测试对象及环境

6.1.1.1 测试对象

基于第四章车控操作系统测试体系的介绍，本节是对安全车控操作系统的测试具体案例的测试及分析。安全车控操作系统测试对象主要面向其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。

对于安全车控系统的测试整体上以实时性、安全性测试为主，其中功能测试主要进行 OS 的基本功能测试、OS 的运行时环境、OS 的基础软件服务测试；性能测试主要进行时间特性的测试、任务调度的性能测试、资源占用测试、网络性能测试；安全测试分为功能安全和信息安全两个维度。

6.1.1.2 测试环境

表 1　软硬件环境和工具

名称	规格/属性	所覆盖的需求
上位机/服务器	Centos	负责整体的测试流程控制和测试结果处理
被测系统硬件环境	嵌入式设备	负责执行测试程序，记录测试结果等

名称	规格/属性	所覆盖的需求
串口线	Micro usb 接口	设备可通过 uart 进行串口 log 查看，方便 debug
网线	千兆以上	设备联网环境
电源	12V 5A	设备供电电源
支持 SSH/telnet 的终端仿真程序	/	用于通过网络访问设备登陆界面
显示客户端	/	用于显示设备输出信息
升级工具	64 位	用于对设备进行升级
系统镜像	/	烧录到设备端的镜像
Canoe 设备	/	用于进行 can 协议相关测试
示波器、稳压电源	/	用于进行 can 一致性等测试
高低温箱	/	用于进行高低温的压力测试

6.1.2 测试流程

测试流程根据第四章的介绍也是从功能测试、性能测试、安全测试三个维度开展。并根据上述三个维度来进行相关测试案例的介绍。

6.1.2.1 功能测试

OS 的功能测试实例如下：

表 2 安全车控 OS 的功能测试实例

序号	测试方法
1	使用 coherent DMA buffer 分配和释放 DMA 内存测试
2	动态申请 dma0 的 4 个 chn，传输数据测试
3	DMA pool 分配和释放小容量的 DMA buffer 测试
4	休眠唤醒之后，dma0 能正常使用 coherent DMA buffer 分配和释放 DMA 内存测试
5	上电 cpu7：指定 1 个中断号在 cpu7 上，例如指定中断号 10 只上报到 cpu7，执行 cpu7 上电命令，连续多次查看测试中断号上报的 cpu core 测试
6	下电 cpu7：指定 1 个中断号在 cpu7 上，例如指定中断号 10 只上报到 cpu7，且指定一个进程运行在 cpu7 上；执行 cpu7 下电命令后，连续多次查看测试中断号上报的 cpu core，以及 top 查看运行的 cpu core 测试
7	cpu1-7 都下电，再全都上电；上下电过程中均连续多次查看系统中断号上报的情况测试
8	两个操作系统核间通讯测试，A 系统和 B 系统之间，或 A 系统和 C 系统之间通路一直相互收发测试
9	A 和 B 两个操作系统，接收 A 的 panic 触发的软件错误，控制 B 重启测试

序号	测试方法
10	A 和 B 两个操作系统，接收 A 的 reboot 命令触发的软件错误，控制 B 重启测试
11	A 系统发生 wdg，panic，心跳丢失的错误时，B 系统会发送 backtrace 命令给 A 系统测试

6.1.2.2 性能测试

OS 的性能测试实例如下：

表 3　安全车控 OS 的性能测试实例

序号	测试方法
1	DMA IO 性能：使用 ion_memcpy 测试 512M 数据搬移，连续测试 100 次
2	时间同步性能：NTP/PTP/GPS 系统启动 RTC 同步时间性能测试
3	时间同步性能：NTP/PTP/GPS 系统唤醒 RTC 同步时间性能测试
4	时间同步性能：以太网 PTP 授时精度性能测试
5	CAN 启动的速度测试
6	CAN 唤醒的速度测试
7	CAN 最高传输速率测试

6.1.2.3 安全测试

安全测试主要分为功能安全和信息安全两大类，这两类安全从流程上会参考第 2 章第 4 节提到的流程标准规范如：ISO 26262、ISO 21434、ASPICE 进行开发和测试。安全车控操作系统对实时性和安全性要求较高，要满足的安全等级一般是 ASIL-D；功能安全测试流程遵循以下 V model，测试整个生命周期主要由以下两类测试子流程组成（软件测试和系统测试），其整体流程如下图所示。

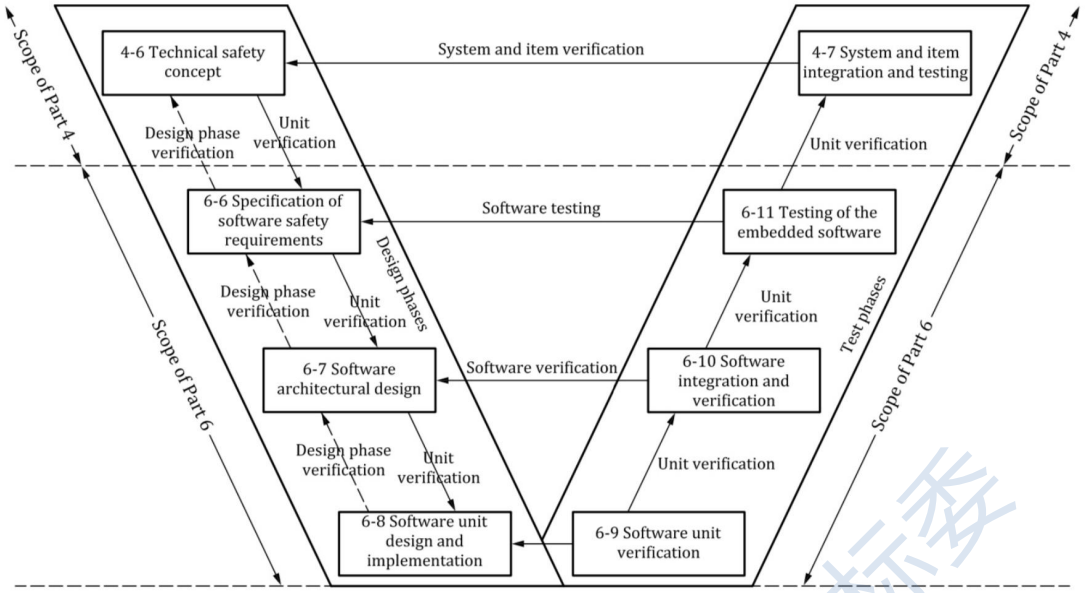

图 17 软件开发的参考阶段模型

流程中软件的单元测试、集成测试、合格性测试，系统测试都必不可少。其中对各类测试的验证结果定义可参考如下规则：

1、总则

表 4 测试结果定义规范

序号	类别	定义
1	测试用例通过占比（百分比）	$\left(Number\ of\ Passed\ Tests / Total\ number\ of\ tests\ executed\right) \times 100$
2	测试用例失败占比（百分比）	$\left(Number\ of\ Failed\ Tests / Total\ number\ of\ test\ executed\right) \times 100$
3	测试用例阻塞占比（百分比）	$\left(Number\ of\ Blocked\ Tests / Total\ number\ of\ test\ executed\right) \times 100$
4	缺陷 SMI 值	$\sum_{k=0}^{n} \binom{n}{k} Coefficient\ of\ bug\ status^{k} * Coefficient\ of\ bug\ serverity^{k}$
5	内存消耗	运行期间的 memory 使用情况
6	CPU 占用率	运行期间的 CPU 使用情况
7	NPU 占用率	运行期间的 NPU 使用情况（AI 处理单元）

2、代码覆盖率

表 5　代码覆盖率定义规范

序号	类别	功能安全 ASIL 等级
1	语句覆盖率	A/B
2	分支覆盖率	B/C/D
3	MC/DC(ModifiedCondition/Decision Coverage)	D
4	函数覆盖率	C/D
5	调用覆盖率	C/D
6	需求覆盖率	A/B/C/D

3、编码规则符合性

表 6　编码规则符合性定义规范

序号	类别	功能安全 ASIL 等级
1	MISRA C 合规指数	A/B/C/D
2	AUTSAR C++顺应性指标	A/B/C/D

4、代码质量

表 7　代码质量定义规范

序号	类别	定义	推荐值
1	STCDN	注释与代码比率	＞0.2
2	SYCYC	循环复杂度	＜15
3	STXLN	可执行行数	＜70
4	STSUB	函数调用数	<10
5	STMIF	最深嵌套数	＜5
6	STPTH	静态代码路径评估	＜250

5、需求覆盖度

表 8　需求覆盖度定义规范

序号	类别	定义
1	需求覆盖度（百分比）	（测试用例涵盖的需求数/所有需求数）*100
2	需求通过率（百分比）	（需求对应的用例通过数/所有需求对应的用例数）*100

6.1.3　测试结果及分析

6.1.3.1 功能测试结果分析

基于 6.1.2.1 节对功能相关测试方法的介绍，本部分主要对这些

测试方法的测试结果及分析的介绍，具体如下（序号和测试方法中的序号一一对应）。

表 9　安全车控 OS 功能测试实例结果分析

序号	测试结果	结果分析
1	测试程序正常执行，log 打印 pass；	PASS
2	测试程序正常执行，1000 次数据 copy 都正确；	PASS
3	测试程序正常执行，log 打印 pass；	PASS
4	系统进入休眠；系统被唤醒；测试程序正常执行，log 打印 pass；	PASS
5	cpu7 上电成功；可以查看到 cpu7 的信息；测试中断号数量在 cpu7 上增加；测试程序运行在 cpu7 上；	PASS
6	测试中断号数量只在 cpu7 上增加；测试进程运行在 cpu7 上；cpu7 下电； 没有 cpu7 的信息；测试中断号在其它 cpu core 上正常增加；测试进程运行在其它的 cpu core 上，且测试进程正常运行；	PASS
7	cpu1-7 都下电成功；只有 cpu0 的信息；只有 cpu0 上的中断数量在增加；cpu1-7 都上电成功；cpu1-7 上的中断号也有增加；cpu1-7 上也有进程；重启成功；查看到 cpu0-7 的信息；cpu0-7 上的中断号有增加；	PASS
8	模拟收发的程序运行成功，不报错	PASS
9	使用工具制造错误，且相关错误能正常上报，上报后有异常恢复机制	PASS
10	使用工具制造错误，且相关错误能正常上报，上报后有异常恢复机制	PASS
11	使用工具制造 wdg，panic，心跳丢失的错误信息，检查 B 系统发送命令成功到 A 系统	PASS

6.1.3.2 性能测试结果分析

基于 6.1.2.2 节对性能相关测试方法的介绍，本部分主要对这些测试方法的测试结果及分析的介绍，具体如下（序号和测试方法中的序号一一对应）。

表 10　安全车控 OS 性能测试实例结果分析

序号	测试结果	结果分析
1	编写相关测试程序，测试结果查看平均吞吐率达到测试指标要求	PASS
2	编写代码实现系统重启后，从网络连接好到 RTC 同步好的时间，符合指标要求测试通过	PASS

序号	测试结果	结果分析
3	编写代码实现系统唤醒后，从网络连接好到 RTC 同步好的时间，符合指标要求测试通过	PASS
4	测试 ptp service 授时精度在性能指标要求的时间范围内	PASS
5	记录从上电到系统发出第一个 can message 的时间，符合指标要求则测试通过	PASS
6	记录从收到 can wake up 信号到系统唤醒后 can 发出第一个包的时间，符合指标要求则测试通过	PASS
7	测试 can 的实际最高传输速率，符合指标要求则测试通过	PASS

6.1.3.3 安全测试结果分析

基于 6.1.2.3 节对安全相关测试方法的介绍，这里针对安全测试方法，主要定义了测试完成的标准，作为结果分析的依据。主要从软件单元测试、软件集成测试、软件合格性测试，系统测试维度来定义。

1、软件单元测试

1）总则

表 11　软件单元测试结果通过标准

序号	类别	规则
1	测试用例通过占比	100%
2	测试用例失败占比	0%
3	测试用例阻塞占比	0%
4	缺陷的 SMI 值	0

2）代码覆盖率

表 12　软件单元测试代码覆盖率通过标准

序号	类别	功能安全 ASIL	规则
1	语句覆盖度	A/B	100%(<100% add rationale)
2	分支覆盖度	B/C/D	90%
3	需求覆盖度	A/B/C/D	100%

3）代码规则符合性

表 13　软件单元测试代码规则符合性通过标准

序号	类别	功能安全 ASIL	规则
1	MISRA C 合规指数	B/C/D	100%(<100% add rationale)
2	AUTSAR C++顺应性指标	B/C/D	100%(<100% add rationale)

4）代码质量

表 14　软件单元测试代码质量通过标准

序号	类别	规则
1	STCDN	> 0.2
2	SYCYC	< 15
3	STXLN	< 70
4	STSUB	<10
5	STMIF	< 5
6	STPTH	< 250

2、软件集成测试

1）总则

表 15　软件集成测试结果通过标准

序号	类别	规则
1	测试用例通过占比	>98%
2	测试用例失败占比	<2%
3	测试用例阻塞占比	0%
4	缺陷 SMI 值	<100 & No major or block bugs

2）需求覆盖度

表 16　软件集成测试需求覆盖度通过标准

序号	类别	规则
1	需求覆盖度	100%
2	需求通过率	>98%

3、软件合格性测试

1）总则

表 17　软件合格性测试结果通过标准

序号	类别	规则
1	测试用例通过占比	>98%
2	测试用例失败占比	<2%
3	测试用例阻塞占比	0%
4	缺陷 SMI 值	<100 & No major or block bugs

2）需求覆盖度

表 18　软件合格性测试需求覆盖度通过标准

序号	类别	规则
1	需求覆盖度	100%
2	需求通过率	>98%

4、系统测试

1）总则

表 19　系统测试结果通过标准

序号	类别	规则
1	测试用例通过占比	>98%
2	测试用例失败占比	<2%
3	测试用例阻塞占比	0%
4	缺陷 SMI 值	<100 & No major or block bugs
5	Memory 消耗	运行期间没有 OOM，在不低于 80%的压力下测试，DDR 的使用稳定
6	NPU 占用率	在不低于 80%的压力下测试，NPU 的使用稳定
7	CPU 占用例	在不低于 80%的压力下测试，CPU 的使用稳定

2）需求覆盖度

表 20　系统测试需求覆盖度通过标准

序号	类别	规则
1	需求覆盖度	100%
2	需求通过率	>98%

6.2 智能驾驶操作系统测试

6.2.1　测试对象及环境

6.2.1.1 测试对象

基于第四章车控操作系统测试体系的介绍，本节是对智能驾驶操作系统的测试具体案例的测试及分析。智能驾驶操作系统的测试对象主要分系统软件和功能软件两大部分，系统软件包括车规级操作系统内核，运行的硬件载体是高性能计算异构芯片，内核通常以兼容标准的 POSIX 接口为基础，支持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。

功能软件是车控操作系统中根据面向服务的架构（SOA）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。

6.2.1.2 测试环境

表 21　软硬件环境和工具

名称	规格/属性	所覆盖的需求
上位机/服务器	Centos	负责整体的测试流程控制和测试结果处理
被测系统硬件环境	嵌入式设备	负责执行测试程序，记录测试结果等
串口线	Micro usb 接口	设备可通过 uart 进行串口 log 查看，方便 debug
网线	千兆以上	设备联网环境
电源	12V 5A	设备供电电源
支持 SSH/telnet 的终端仿真程序	/	用于通过网络访问设备登陆界面
显示客户端	/	用于显示设备输出信息
升级工具	64 位	用于对设备进行升级
系统镜像	/	烧录到设备端的镜像
高低温箱	/	用于进行高低温的压力测试
Canoe 设备	/	用于进行 can 协议相关测试
示波器、稳压电源	/	用于进行 can 一致性等测试
光学实验室环境	/	智能驾驶视觉图像效果客观测试
实车环境	/	智能驾驶视觉图像效果主观测试、功能测试、实车测试
仿真台架	/	用于仿真测试需要
软件在环环境	/	用于仿真测试需要
硬件在环环境	/	用于仿真测试需要
仿真可视化工具	/	用于仿真测试需要

6.2.2　测试工具

以 Linux 为例，通常使用如下工具进行 linux 内核相关的测试。

表 22　常见的 Linux 内核测试工具

工具名称	测试对象
lmbench	测试文件读写、内存操作、进程创建销毁开销、网络等性能
Unixbench、ftrace、Linpack	内核性能测试

工具名称	测试对象
rt-tests	内核实时性测试
Hackbench	进程调度性能、资源占用测试
Dbench、debugfs	文件系统负载
spec2000	CPU 性能、稳定性测试
stressapptest	内存稳定性测试
memtester	内存压力测试
stream、ubench	内存带宽测试
iozonetest3a、dd、bonnie++	磁盘 IO 性能、稳定性测试
Lmbench	综合性能测试
iperf、netperf	网络性能测试
BPF	内核观测（在内核 4.9 版本后引入）

6.2.3 测试流程

对于智能驾驶操作系统的开发设计和测试整体上遵循 ASPICE 标准的 V 模型，测试整个生命周期主要由以下两类测试子流程组成（软件测试和系统测试），其整体流程如下图所示。

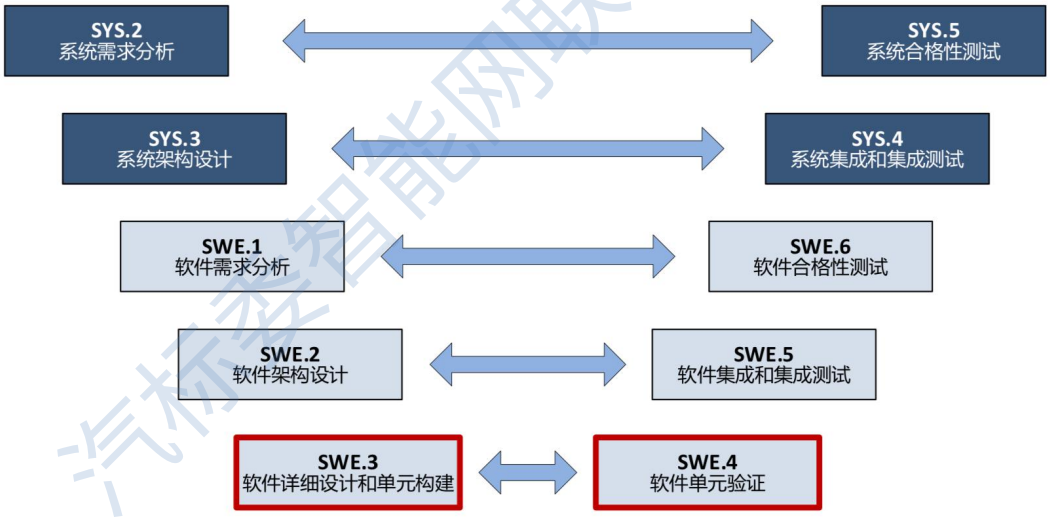

图 18 ASPICE 标准 V 模型

结合第四章的介绍，从测试类别维度来分，智能驾驶操作系统测试又可分为功能、性能、安全测试三个维度。其中，功能测试分为针对 OS 的基础功能测试、应用框架基础功能测试；性能测试主要是可靠性测试、时间特性测试、I/O 性能测试、资源占用测试、网络性能

测试；安全测试分为功能安全和信息安全两个维度。

下面将根据这三个维度来进行相关测试的案例介绍。

6.2.3.1 功能测试

（1）OS 的基础功能测试

1）文件管理功能测试案例：

以文件存储相关测试为例，常用测试方法如下：

表 23　智能驾驶 OS 功能之文件管理测试实例

序号	测试方法
1	userdata 分区下进行读写操作测试
2	格式化 app 分区后再次读写操作测试
3	不同类型 TF 卡的读写、热插拔、擦除测试
4	TF 卡、分区的挂载测试
5	不同 flash 反复读写擦除测试（emmc/hyper/nor/nand）
6	对 flash 擦除超过实际分区大小的测试
7	对 flash 数据读写，设置传输数据大小超过支持范围的测试
8	对 flash 数据读写，读写过程中 kill 进程，之后再启动数据读写测试
9	对 flash 数据读写，读写过程中，系统重启（之后再启动数据交换）测试
10	对系统调用休眠唤醒后，进行数据读写测试

2）网络管理功能测试案例：

以 Ethernet 为例，常用测试方法如下：

表 24　智能驾驶 OS 功能之网络管理（Ethernet）测试实例

序号	测试方法
1	反复查看 ethernet 中断信息测试
2	反复查看 ethernet 网口信息测试
3	1000M 速率的通信测试
4	断电重启后 ping 同一网段的设备测试
5	reset 重启后 ping 同一网段的设备测试
6	reboot 重启后 ping 同一网段的设备测试
7	Eth driver 支持帧抢占测试（FP 测试）
8	Eth driver 支持指定时隙发送测试（EST 测试）
9	Eth driver 支持保留带宽,AVB-CBS 设定该流所走的 queue 的带宽测试
10	IP 协议测试
11	TCP 协议测试
12	UDP 协议测试

序号	测试方法
13	网卡 down 和 up 测试
14	配置、调整并获取 Ethernet MAC 内置 timestamp 时间测试

以 CAN 为例，常用的功能测试方法如下：

表 25　智能驾驶 OS 功能之网络管理（CAN）常用功能测试实例

序号	测试方法
1	CAN 协议注册查询、网络设备节点检查测试
2	CAN 中断信息反复查看测试
3	CAN 基本功能测试-1 台被测设备 4 个 can 之间的通信
4	CAN 基本功能测试-2 台被测设备 4 个 can 之间的通信
5	CAN 基本功能测试-1 台被测设备 2 个 can 之间的通信
6	CAN 基本功能测试-同时发送不同 can_id 的不同数据测试
7	CAN 接收信息过滤测试——只接收 1 个 can_id 的信息
8	CAN 接收信息过滤测试——只接收 2 个 can_id 的信息
9	CAN speed 测试——设置接收与发送的传输速率不一致
10	CAN speed 测试——设置最大传输速率测试
11	CAN speed 测试——设置超出最大传输速率的测试
12	监听模式下测试 CAN 发送接收
13	CAN 支持系统唤醒（外部设备唤醒）测试
14	CAN 支持系统唤醒，两个设备的 can0 相互唤醒测试
15	CAN 多路测试

以 CAN 为例，主要的物理层一致性测试方法如下：

表 26　智能驾驶 OS 功能之网络管理（CAN）物理层一致性测试实例

序号	测试方法
1	CAN 总线电压测试-总线输出电压测试
2	CAN 总线电压测试-总线输入电压限值测试
3	内阻测试
4	边沿测试-最小负载测试
5	位时间精度测试
6	信号对称性测试
7	采样点测试
8	故障管理-DUT 掉电/掉地/短路/断路测试
9	故障管理-地偏移测试
10	通信电压-通信电压范围测试
11	通信电压-欠压/过压情况下报文发送持续时间测试
12	位宽容忍度测试
13	扩展报文帧的兼容性测试

（2）应用框架基础功能测试

1）升级功能以 OTA 为例，典型测试方法如下：

表 27　智能驾驶 OS 功能之 OTA 升级功能测试实例

序号	测试方法
1	分区升级：从低版本升级到高版本（EMMC\gloden\AB 模式）测试
2	整包升级：从低版本升级到高版本（EMMC\gloden\AB 模式）测试
3	差分包升级：从低版本到高版本测试
4	差分包升级：从高版本到低版本测试
5	单分区 boot 分区差分包升级：从低版本升级到高版本测试
6	单分区 boot 分区差分包升级：从高版本升级到低版本测试
7	单分区 system 分区差分包升级：从低版本升级到高版本测试
8	单分区 system 分区差分包升级：从高版本升级到低版本测试
9	篡改分区信息后升级失败测试
10	带签名信息的 ota 升级测试
11	篡改签名信息后升级失败
12	升级过程中断电测试

6.2.3.2 性能测试

（1）可靠性测试

举例：系统 DDR 压力测试

测试方法：使用 linux 开源工具 stressapptest 进行内存压力测试：测试内存 100MB，2 个临时文件增加写磁盘文件的线程，运行内存翻转线程设置为 4，运行内存拷贝线程数设置为 8，运行 CPU 压力线程数设置为 2，使用更高 CPU 负载的内存拷贝测试；同时，通过 NPU 双核运行大压力模型进行压力测试，使 NPU 双核占比达到 90%以上；最终使 CPU，NPU 占比均接近 100%，同时 DDR 占比接近实际最大带宽。

测试时长：10 台设备，每台运行 72h。

测试环境：常温、高温、低温、温度循环。

（2）网络性能测试

举例：ethernet 性能测试，以千兆以太网为例

测试方法：使用 linux 开源工具 iperf，进行 TCP 的带宽性能测试：发送性能，将 PC 作为 server 端，被测设备作为 client 端，运行时长 60 秒，记录性能结果；接收性能，将被测设备作为 server 端，PC 作为 client 端，运行时长 60 秒，记录测试结果。

6.2.3.3 安全测试

安全测试主要分为功能安全和信息安全两大类，这两类安全从流程上会参考第 2 章第 4 节提到的流程标准规范如：ISO 26262、ISO 21434、ASPICE 进行开发和测试。智能驾驶操作系统要满足的安全等级一般是 ASIL-B/ASIL-D。相关的流程和规范定义和 6.1.2.3 章节基本保持一致，差异之处在于 ASIL-B/D 级别要求上的差异。

6.2.4 测试结果及分析

6.2.4.1 功能测试结果及分析

（1）OS 的基础功能测试

基于 6.2.2.1 节对功能相关测试方法的介绍，本部分主要对这些测试方法的测试结果及分析的介绍，具体如下（序号和测试方法中的序号一一对应）。

1）文件管理功能测试案例

表 28 文件管理测试实例结果分析

序号	测试结果	结果分析
1	系统重启后文件 md5 值不变，/userdata 分区总大小不变	PASS
2	/app 目录被清空，系统重启后文件 md5 值不变，/app 分区总大小不变	PASS

序号	测试结果	结果分析
3	TF 卡的文件系统格式正确；文件、文件夹创建成功；文件拷贝成功；热插拔 TF 卡后，TF 卡信息正常，文件 md5 值不变；擦除后，读写正常	PASS
4	正常挂载成功，无报错	PASS
5	可以看到分区大小；flash 擦除没有报错；flash 读写成功，且文件的 MD5 一致	PASS
6	擦除失败，报错 MEMERASE: Invalid argument	PASS
7	数据擦除与写入正常，读取数据时只能读取 mtd21\22 分区大小的数据，日志提示相应错误信息	PASS
8	擦除、写入、读取进程被 kill 后，再次执行，均正常，无报错	PASS
9	擦除、写入、读取进程被中断后，重启后再次执行，均正常，无报错	PASS
10	读写正常，无 error/failed/timeout 报错	PASS

2）网络管理功能测试案例：

表 29　网络管理测试实例结果分析

序号	测试结果	结果分析
1	可以成功查到 eth 的中断信息，且中断值不断增加	PASS
2	可以成功查到 eth0 和 eth1 的网口信息	PASS
3	设置速率成功，可以 ping 通，关注发送/接收的时延级别 ms	PASS
4	每次均能 ping 通；关注发送/接收的时延级别 ms；允许丢包个数不大于 1 个	PASS
5	每次均能 ping 通；关注发送/接收的时延级别 ms；允许丢包个数不大于 1 个	PASS
6	每次均能 ping 通；关注发送/接收的时延级别 ms；允许丢包个数不大于 1 个	PASS
7	设置 est 成功（status: enabled）；tperf 只能发送优先级 3 的数据，优先级 1、2 的数据不能发出；I/O 图表中连续 1s 内大约有 10 个数据记录	PASS
8	期望 q1 的带宽不超过 1G 的 10%，即 102.4Mbps；期望 q2 的带宽不超过 1G 的 25%，即 256Mbps；期望 q3 的带宽不超过 1G 的 40%，即 409.6Mbps	PASS
9	dmesg 可以找到 IP 的注册日志 测试设备可以 ping 通 PC 的 IP	PASS
10	dmesg 可以找到 tcp 的注册日志；telnet 登录设备 IP 成功(输入 root 即可),并成功进入/userdata 下	PASS
11	dmesg 可以找到 udp 的注册日志；远端磁盘挂载成功，进入 mnt 下，可以正常读写文件等	PASS
12	每次均能 ping 通；关注发送/接收的时延级别 ms	PASS
13	设置 PTP 时间成功；获取 PTP 时间成功；调整 PTP 时间成功	PASS

（2）应用框架基础功能测试

1）升级功能以 OTA 为例，典型测试方法如下：

表 30　OTA 升级功能测试实例结果分析

序号	测试结果	结果分析
1	查看升级状态，cat/userdata/log/cache/result，值为 1 表示升级成功	PASS
2	查看升级状态正确性，值为 1 表示升级成功；查看版本号的正确性；检查 lib 文件的 md5 值，/system/lib 中 so 文件与 AppSDK 包中 lib 文件的 MD5 值比较一致性	PASS
3	查看版本号的正确性；查看升级状态，cat/userdata/log/cache/result，值为 1 表示升级成功	PASS
4	查看升级状态，cat/userdata/log/cache/result，值为 2 表示升级失败	PASS
5	查看版本号的正确性；查看升级状态，cat/userdata/log/cache/result，值为 1 表示升级成功	PASS
6	查看升级状态，cat/userdata/log/cache/result，值为 2 表示升级失败	PASS
7	查看版本号的正确性；查看升级状态，cat/userdata/log/cache/result，值为 1 表示升级成功	PASS
8	查看升级状态，cat/userdata/log/cache/result，值为 2 表示升级失败	PASS
9	查看升级状态，cat/userdata/log/cache/result，值为 2 表示升级失败	PASS
10	查看升级状态，cat/userdata/log/cache/result，值为 1 表示升级成功；查看版本号的正确性；检查 lib 文件的 md5 值	PASS
11	查看升级状态，cat/userdata/log/cache/result，值为 2 表示升级失败	PASS
12	每次断电后，升级失败，重新上电后进入 recovery 模式 cat/userdata/log/cache/result，显示值为 2；最后一次升级成功 cat/userdata/log/cache/result，显示值为 1	PASS

6.2.4.2 性能测试结果及分析

基于 6.2.2.2 节对性能相关测试方法的介绍，本部分主要对这些测试方法的测试结果及分析的介绍，具体如下（序号和测试方法中的序号一一对应）。

（1）可靠性测试结果及分析

在测试过程中，如果出现"error"或者"miscompare"关键字，也说明测试失败。测试过程中出现 shedule_work fail 或者 pausing worker threads 是正常的。

比如下图测试显示 pass，代表 cpu 测试通过。

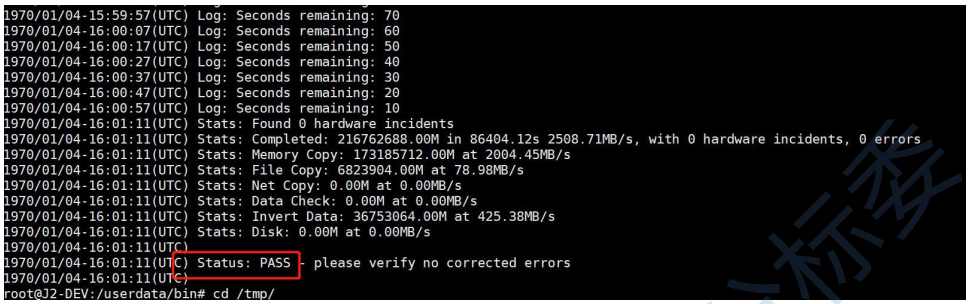

图 19 CPU 压力测试通过图例

下图显示 fail，代表 cpu 测试不通过。

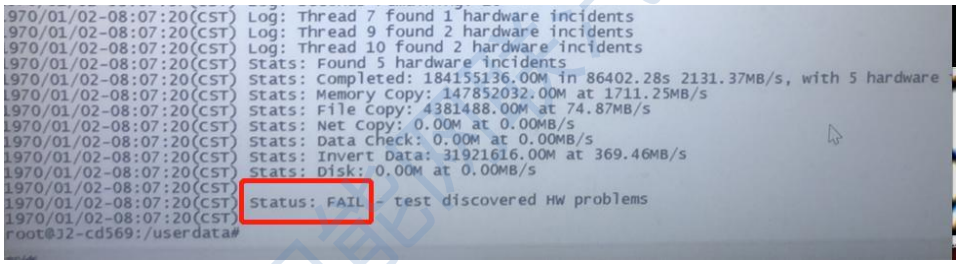

图 20 CPU 压力测试不通过图例

以上在常温、高温、低温、温度循环四种环境下均测试成功，则代表测试通过。

（2）网络性能测试结果及分析

以千兆以太网为例，假如测试通过的标准为发送带宽：870Mbits/sec；接收带宽：940Mbits/sec；如果发送带宽测试结果大于870Mbits/sec，则测试通过，否则测试失败；如果接收带宽测试结果大于 940Mbits/sec，则测试通过，否则测试失败。

6.2.4.3 安全测试结果及分析

智能驾驶操作系统的安全测试结果及分析和 6.1.3.3 基本保持一致。差异之处在于 ASIL-B/D 级别要求上的差异。

6.3 车载操作系统测试

6.3.1 测试对象及环境

测试对象：某车载导航娱乐系统

测试环境：参见下图

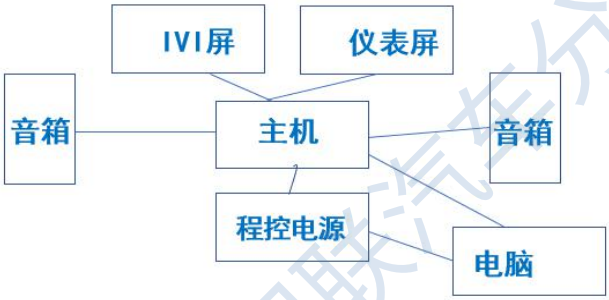

图 21 某车载导航娱乐系统测试环境

6.3.2 测试流程

这里列举针对车载导航娱乐系统的时延指标测试，测试流程如下：

（1）通过高帧率相机录制视频，进行测试步骤操作，根据时延指标起止点数帧计算各时延指标；

（2）通过秒表计时获取时间或用示波器看周期。

6.3.3 测试结果及分析

表 31 某车载导航娱乐系统测试结果及分析-案例 1

测试标题	RVC 启动完成时延
测试描述	激活 RVC 开始，到 RVC 界面稳定显示为 RVC 启动完成时延。

测试标题	RVC 启动完成时延
	RVC 启动完成时延数值越小越好。
预置条件	1) 车机处于开机状态
	2) 此次开机未启动过 RVC
测试步骤	1) 激活 RVC 开始计时；
	2) RVC 界面稳定显示停止计时；
	3) 使用高帧率相机（240FPS）拍摄数帧计时；
	4) 计算 5 次结果的平均值作为最后的结果。
结果分级	RVC 启动完成时延：
	A：T <= 0.5s　B：0.5s < T <=1s　C：T > 1s
备注	稳定显示指连续多帧画面不发生变化取最前面一帧

表 32　某车载导航娱乐系统测试结果及分析-案例 2

测试标题	QNX+Android 系统启动完成时延
测试描述	ACC 上电开始计时，到系统全部完成启动为 QNX+Android 系统启动完成时延
	QNX+Android 系统启动完成时延数值越小越好。
预置条件	1) ACC OFF 进入休眠状态
测试步骤	1) ACC 上电开始计时
	2) 系统全部完成启动稳定显示停止计时；
	3) 使用高帧率相机（240FPS）拍摄数帧计时；
	4) 计算 5 次结果的平均值作为最后的结果。
结果分级	QNX+Android 系统启动完成时延：
	A：T <= 14s　B：14s < T <= 20s　C：T　> 20s
备注	稳定显示指连续多帧画面不发生变化取最前面一帧

7　标准化建议

车用操作系统测试是智能网联汽车功能、性能与安全的重要保障，是决定用户体验的重要环节。标准体系的不统一、功能定义的不一致，严重影响了消费者对产品的认知和接受度，制约了智能网联汽车产业的发展。亟需通过操作系统的标准化实现产业链协同合作，构建协同演进的行业生态圈，现提出如下标准化建议：

表 33　标准化项目建议

标准化对象	必要性分析	启动建议
车控操作系统试验方法	通过规定车控操作系统的功能、性能、信息安全、功能安全等内容的试验方法，保证产品的质量。	优先级高
车载操作系统试验方法	通过规定车载操作系统的功能、性能、信息安全、功能安全等内容的试验方法，保证产品的质量。	优先级高
车用可信执行环境操作系统技术要求与试验方法	将敏感数据存储在可信执行环境操作系统中，在底层与富执行环境操作系统（安卓等）进行硬件隔离，通过多线程通讯获得必要的信息，由此保证汽车软件的底层安全。	优先级较高