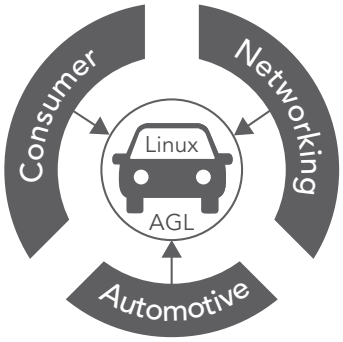
WIND RIVER

Linux for In-Vehicle Systems: A Strategic Platform

Table of Contents

Executive Summary 1
Introduction 1
Linux for Automotive Defined 2
OSS, Commercial Software, and Services 3

Executive Summary



While there is no consensus on the definition of Linux for automotive, this white paper will examine and summarize the state of available options for its key technologies. Available off-the-shelf components and solutions from both community-based open source software (OSS) and commercial software products are evaluated against the actual real-world requirements of device original equipment manufacturers (OEMs). This paper also focuses on the need for integration to create a viable Linux for automotive platform and highlights the challenges arising from factors such as code size, project and technology maturity, software licensing, and other IP considerations and presents cost-effective options for OEMs to realize their design goals for automotive Linux systems.

This white paper lays the foundation for defining a Linux for automotive platform, to meet requirements emerging from industry trends and from specific technical needs of the companies involved in building real-world systems.

Introduction

In the past five years the Linux operating system and related open source software have made impressive inroads in embedded systems deployment and have supported both evolutionary and revolutionary changes in intelligent device design practices. Today one-quarter to one-third of 32-bit and 64-bit designs are developed and deployed on Linux-based platforms. That share continues to expand, driven by trends that include the following:

- Convergence of diverse functions into increasingly complex devices
- Software content (lines of code) and application complexity that grow dramatically year over year
- Requirements for embedded application capabilities and connectivity that meet and exceed those for desktop usage
- Original equipment manufacturers' need for supporting multiple CPU architectures or hardware platforms
- Development and implementation of end-to-end applications with common APIs and protocols at every node of infrastructure and in diverse types of client devices
- Inability of single-vendor proprietary software platforms to match the pace of innovation driven by convergence of traditional IT, consumer electronics, the Web, and such specific application areas as automotive

Different market segments and applications areas have adopted Linux and OSS at different rates. Early adopters in networking and communications infrastructure leveraged Linux interoperability with legacy UNIX systems and strong support for standards-compliant networking. Subsequent generations of developers adopted Linux in consumer electronics, mobile telephony, and elsewhere as a platform for innovation; in particular, global OEMs look to Linux for its flexibility, brand-neutrality, reliability, and community-driven rapid pace of evolution.

Today functionality of in-vehicle systems increasingly converges with other intelligent device types and capabilities from the broader fields of consumer electronics, mobile telephony, and even enterprise IT. This convergence in particular creates a need for a platform that can specifically address the following:

- Emerging requirements to meet both automotive and consumer electronics industry standards
- Soaring budgets for test and QA, which will continue to rise as vehicle designs accrue features and functions
- Mismatches in development and product life cycles for consumer and automotive segments
- Aftermarket and channels exerting pressure on OEMs to contribute value to integrated solutions
- Acceleration of innovation and optimized profit margins for OEMs despite long automotive product life cycles

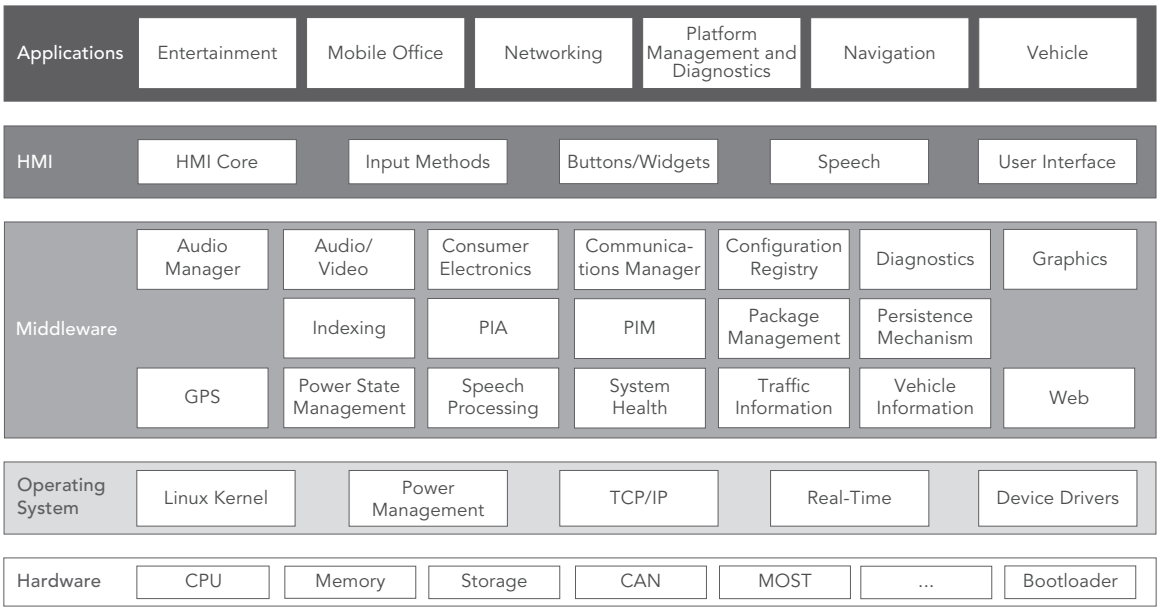


Figure 1: Key Technologies in Linux for Automotive Platform

Linux for Automotive Defined

The requirements of automotive systems OEMs represent a convergence of needs from several distinct segments:

- User-experience and media support from consumer electronics
- Connectivity from data networking, mobile telephony, and traditional automotive systems design
- Performance and reliability from industrial control, infrastructure, and elsewhere

Platform Attributes

To solve these needs, an open source platform must have distinct functional attributes:

- Overall software hardening for improved reliability and performance
- Seamless integration of multimedia, vehicle/traffic information, and telephony
- Support for connectivity within a vehicle, among vehicles (V2V) and with external data sources and roadside infrastructure (V2R)
- Small footprint and fast boot time
- Real-time responsiveness
- Energy, power, and thermal management for CPUs, peripherals, and displays

Platform Architecture

Linux for automotive has implications for every layer of the software stack, from hardware all the way up through value-added applications:

Hardware: Linux for automotive must support both the CPU families of choice for in-vehicle systems (ARM, x86, etc.) as well as specific chipsets/SoCs and associated peripherals supplied by semiconductor manufacturers in reference hardware designs.

Operating system: Linux for automotive must include an up-to-date Linux kernel with device drivers for specific automotive interfaces (e.g., CAN, MOST) as well as popular CODECs, MACs, and protocol stacks from consumer electronics and networking domains. An automotive Linux operating system must also support advanced infrastructure for such features as power state management.

Middleware: To enable current and next-generation-rich applications and user experiences, Linux for automotive must include middleware that spans the gamut for both consumer electronics and mobile telephony while also integrating vehicle control and diagnostics, global positioning, and mapping.

Human-machine interface: The in-vehicle environment presents a range of user-experience requirements and unique modes of use. Linux for automotive must support appropriate input methods (speech, touch, etc.) and user interface capabilities to support both driver and passenger interaction in safety-critical situations with cabin noise, motion, and other challenges without being a distraction itself.

Applications: Ultimately, value-added applications will differentiate in-vehicle systems for both OEM and after-market deployment. A Linux for automotive platform must offer automotive systems OEMs the ability to add new applications, carry forward legacy code, and customize and brand off-the-shelf and included application code.

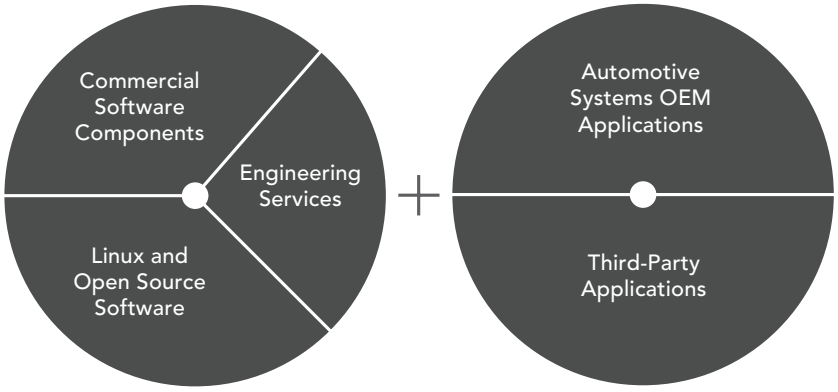


Figure 2: Commercial Approach to Open Source Automotive Solutions

Key Technology Support

Linux for automotive delivers key technologies required for today's, and tomorrow's, in-vehicle platforms:

- Power state management
- Fast boot/init (instant on, standby)
- Media support for key audio, video, and image formats and CODECs
- Graphical output and user interface construction and deployment tools
- Consumer electronics connectivity (USB, wireless)
- Automotive connectivity (MOST, CAN)
- File systems supporting flash memory, RAM, rotating media, and networked storage
- System infrastructure

OSS, Commercial Software, and Services

GNU/Linux and related open source technology represent the foundation of a Linux for automotive platform but require the addition of a sensible mix of complementary commercial software and services to meet the need of OEMs for advanced automotive support.

Community-developed OSS can provide a rich and varied toolkit for developers targeting automotive applications. In some cases, OSS "as is" offers sufficient functionality to meet automotive requirements; in other cases, OEMs are better served by commercially integrated and supported code.

Rapidly evolving community-based software technology often needs additional capabilities to meet exacting automotive requirements. These capabilities can include CPU support, device drivers, standards compliance, protocol support, APIs, quality assurance, and so on. OEMs can acquire these incremental capabilities through internal investment in engineering resources or by partnering with third parties to "bridge the gaps" with commercial software and professional engineering services.