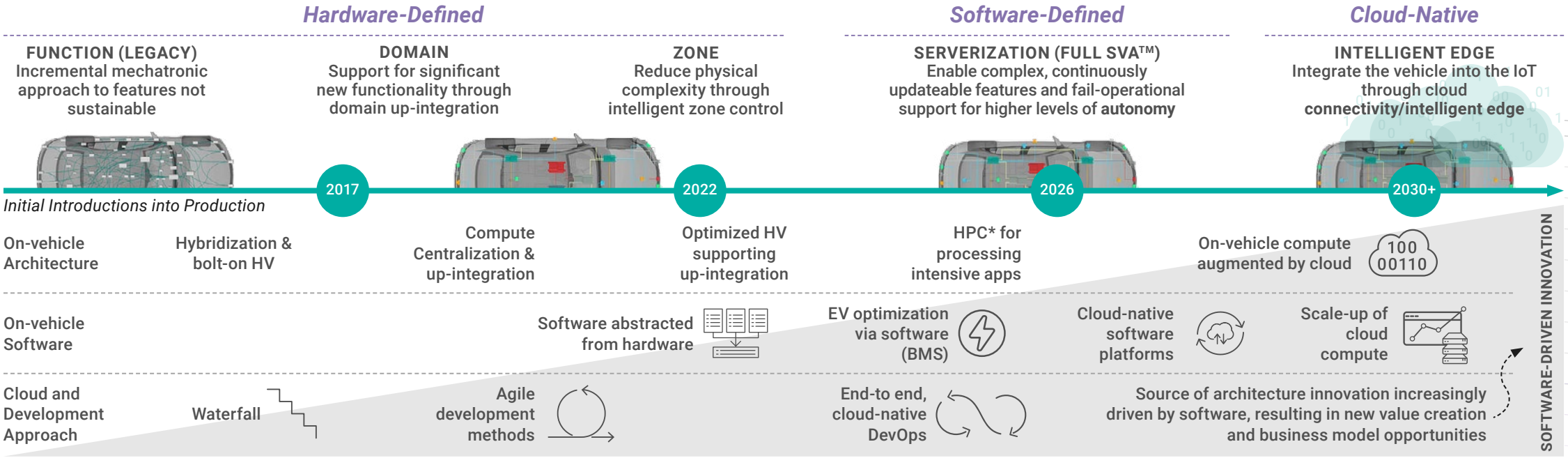
/Vehicle Architecture Evolution Demands Cloud-Ready ECUs

The future is here for automotive software and electronics, according to McKinsey & Co., with software-defined vehicle architecture fast becoming reality, aided by the disruptive force of autonomous driving, connected vehicles, electrification of the powertrain, and shared mobility (ACES). The consulting firm predicts that by 2030 — in about three to four vehicle generations — 70% of vehicles will have a software-defined architecture, and automotive companies across the value chain must begin harnessing its potential now.

Architectural Evolution

/ Vehicle architectural evolution is best understood by looking at milestones, as shown in **Figure 1**. Before 2017, classically driven vehicles were characterized by discrete electronic control units (ECUs) and their embedded software distributed throughout the car. These parts from various vendors were essentially assembled by the OEM with little value addition from the software perspective. If an ECU malfunctioned or required an update, manual replacement was required by qualified engineers at dealerships.



*HPC: High-performance compute, e.g., Aptiv's Open Server Platform

Figure 1: Vehicle E/E architecture is evolving from legacy fixed-function or monolithic through domain and zonal to serverized and cloud-native goals.

Architectural Evolution

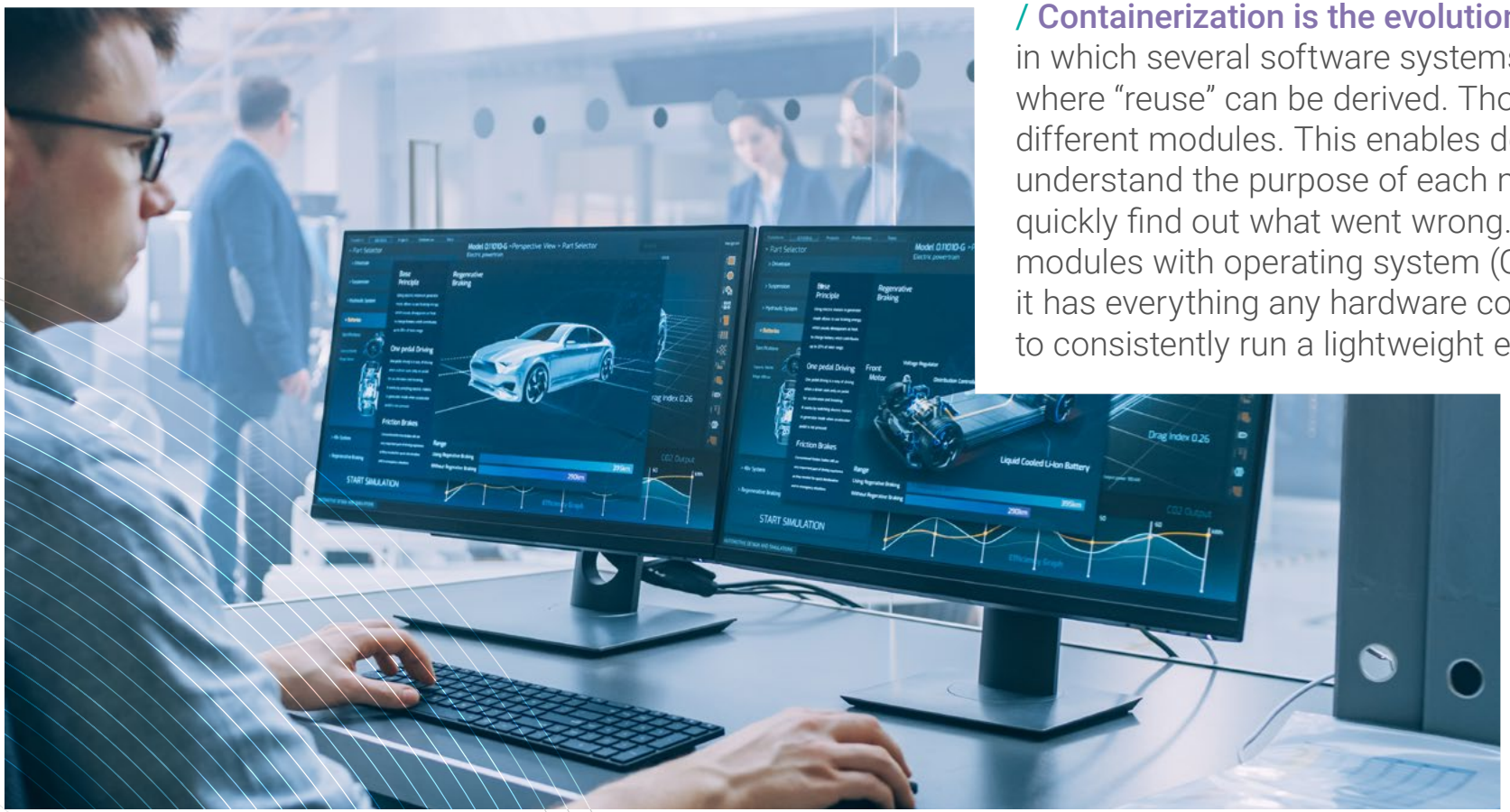
/ By 2017, however, OEMs had begun thinking in terms of domains, such as body control, powertrain, and infotainment, with some fashion of compute centralization. And this domain notion has now evolved toward a cross-domain physical zone based electrical/electronic (E/E) architecture that up-integrates the large number of discrete ECUs to utilize fewer, localized, higher-performance compute systems.

The reason behind this shift is the desire to reduce cabling, weight, and complexity while improving efficiency. However, the ability to accomplish more by defining more elements in software has opened to the door to yet increasing complexity. OEMs continue to extend their zonal architecture toward serverization, leading companies such as Aptiv to further reduce the number of compute units by integrating more software onto larger, high-performance compute (HPC) units.

The cloud-enabled car of 2030 will continue performing the many critical real-time functions onboard, such as antilock braking system code. Rather, the cloud will instead assist via public or private cloud setups in providing additional services such as routing to available parking based on current location or destination.

The E/E architectural evolution has already come a long way on the timeline in Figure 1 and is now beginning to use an agile development model. Over the next few years, OEMs will adopt a software development and IT operations (DevOPs) type of environment that enables continuous integration, continuous test, and continuous delivery. The heavy lifting for this ability will be done by containers.

What is Containerization?



/ Containerization is the evolutionary step up from modularization, in which several software systems are examined to identify areas where “reuse” can be derived. Those areas are then isolated into different modules. This enables development teams to clearly understand the purpose of each module and, if the system fails, quickly find out what went wrong. The container packages such modules with operating system (OS) libraries and dependencies, so it has everything any hardware compute environment would require to consistently run a lightweight executable.

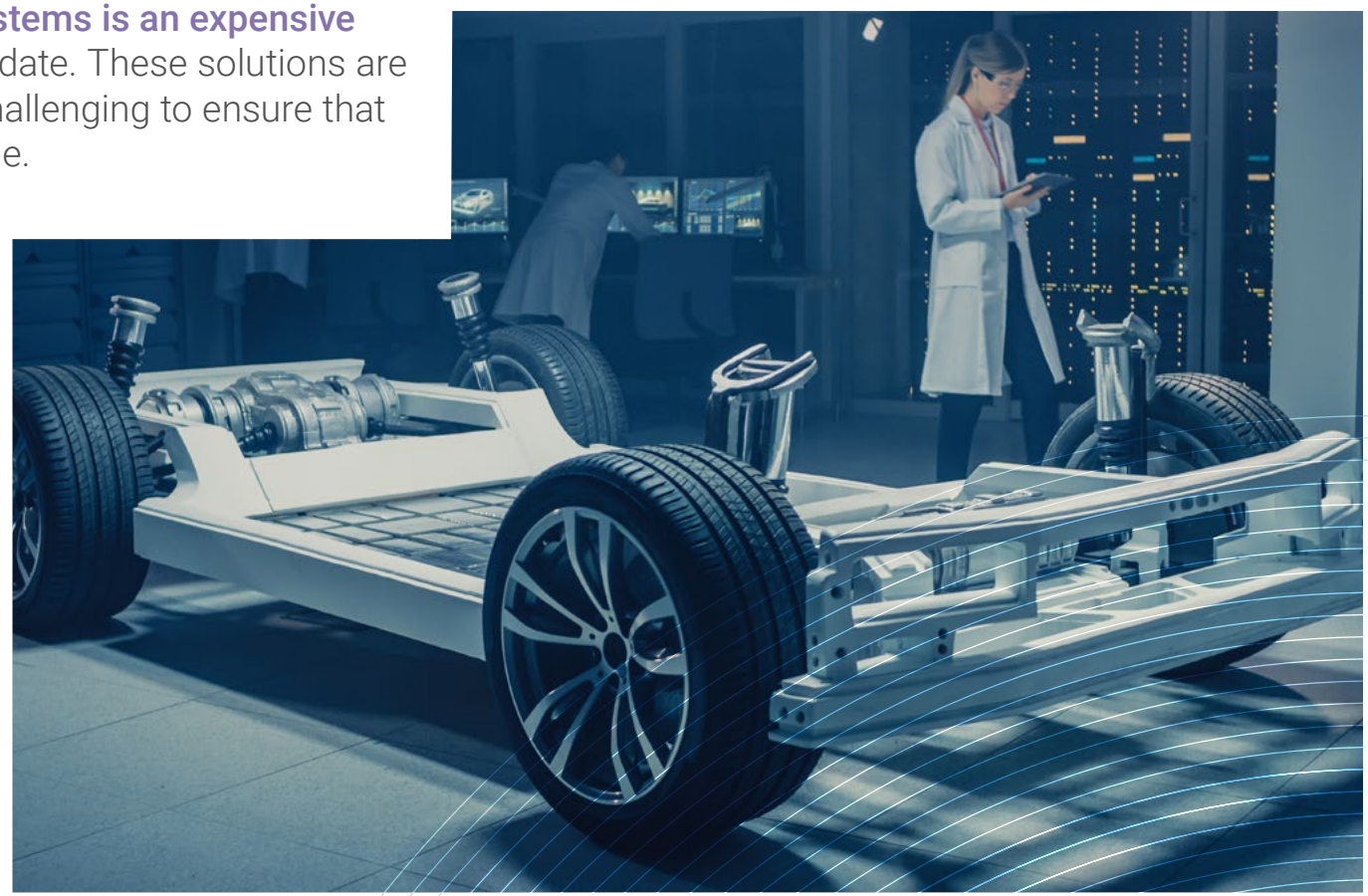
The up-integration of cloud computing technologies into vehicles to provide new features, services, and experiences starts with eliminating existing ECUs by abstracting their functionality, modularizing, and containerizing the modules for delivery.

Containers Ease Management

/ Traditional software management in embedded systems is an expensive task with possibly dozens of software solutions to update. These solutions are monolithic and often proprietary in nature, and it is challenging to ensure that they will reliably work when implemented in the vehicle.

In contrast, the self-contained package that is the container is supported by the Open Container Initiative (OCI)–based runtime interface that makes the deployment runtime-agnostic. The OCI specifies how containers move from the lab to a public or private container registry from which cloud-ready vehicles get access within a security framework. Finally, a tool is needed for managing container delivery from the cloud to the vehicle edge. The Kubernetes open source container orchestration system, widely used by cloud software developers, is the obvious path for this step.

Containerization is thus the model that allows easy management of individual service-oriented blocks of code. Since more than 60% of back end developers already use containers to build and deploy software,2 it is the logical and essential element of the software-first approach to vehicle architecture.



Lift and Shift Design

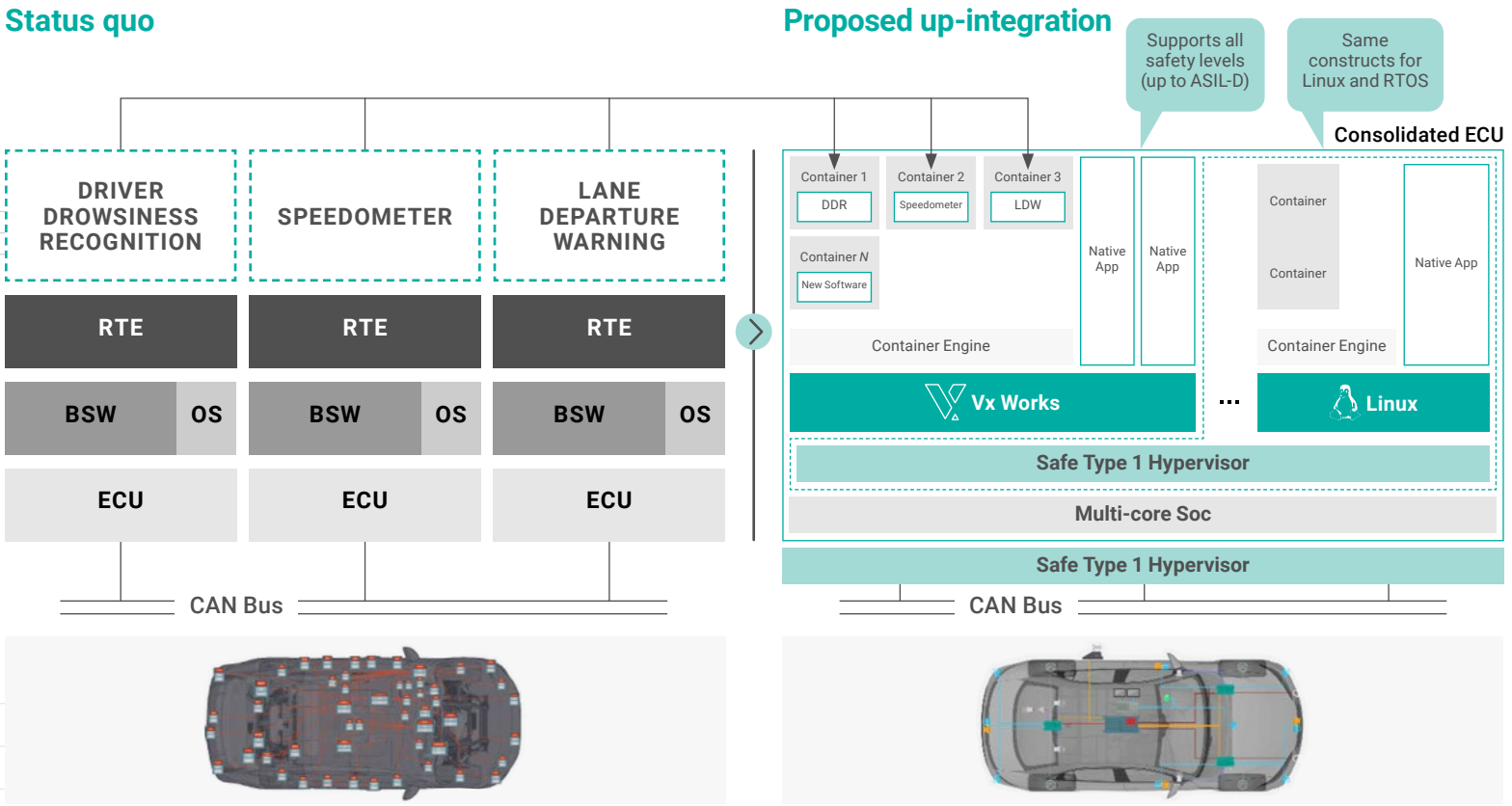


Figure 2: As the only RTOS that supports OCI containers at all levels of safety up to ASIL-D, VxWorks allows software to be containerized and easily Lifted and Shifted to new hardware.

/ The traditional approach to ECU design is fast becoming unsustainable. Take for instance, an existing AUTomotive Open System ARchitecture (AUTOSAR) environment as shown in **Figure 2**, with the real-time environment (RTE), basic software (BSW), and OS layers sitting on the ECU hardware. Each feature functionality, such as driver drowsiness recognition (DDR), lane departure warning, or the speedometer, is implemented monolithically on an independent ECU. This architecture makes it difficult to maintain and upgrade the system due to its inflexible approach — an anathema to progress toward cloud-ready vehicles

Lift and Shift Design

/ In a cloud-ready environment, a Lift-and-Shift or rehosting approach reduces the effort and cost inherent to migration processes. It simply takes an exact application copy, complete in its container, from its legacy environment or the cloud to its new up-integrated environment of multi-core SoC-based HPC.

Engineers can implement Lift and Shift in one of two ways shown in Figure 2 (right). They can lift AUTOSAR stacks packaged in containers and deploy them with a container engine that runs on VxWorks®, which sits on a hosted Type 1 hypervisor. Alternatively, the application, the OS, and the hypervisor can be containerized to make the run environment even more predictable.

Lift and Shift Design

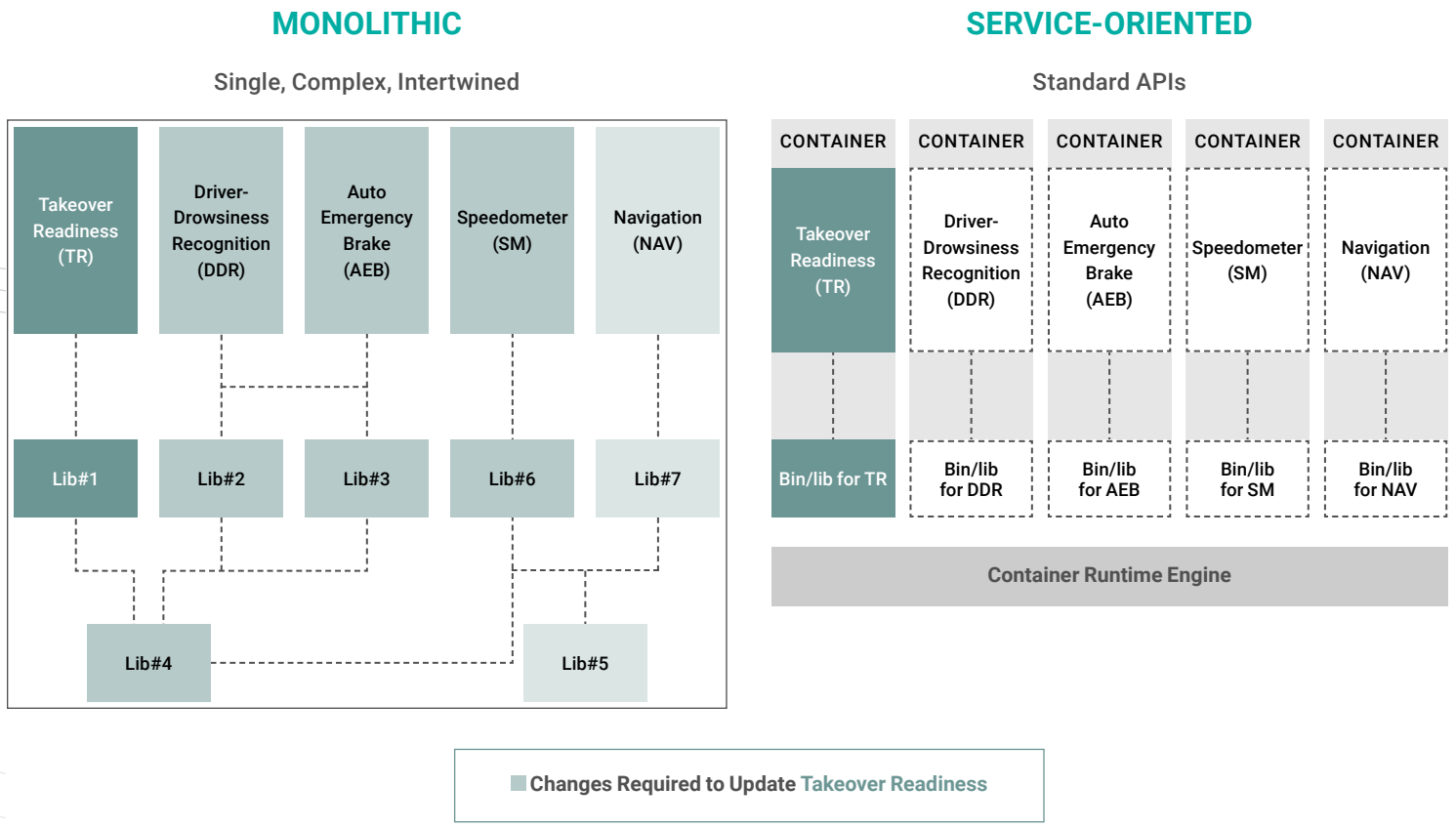


Figure 3: The service-oriented structure eases updates and reuse by avoiding the monolithic structure's dependencies across components.

/ The first benefit of lift and shift, that of software reusability, is illustrated in **Figure 3.** Since there are significant dependencies across software components in the legacy monolithic software structure, code change in one component may require cascading updates across the entire code base. The service-oriented structure's containerized approach, on the other hand, reduces dependencies and code change is contained to only the relevant components.

Architecture Considerations for Evolution

/ There are, however, hardware and software considerations that influence both the type of implementation and the pace of change. Hardware considerations include:

Cost

Existing microcontrollers in ECUs are cheap compared to the SoCs needed for HPC. Although HPC units needed are fewer, the trade-off between the competitive capabilities desired and the cost they could entail must therefore be carefully evaluated.

Power consumption

With the electrification of the power train in mind, it is important to consider the power budget through a comparison of the total power consumed by numerous microcontrollers (lower power, less efficient) with fewer SoCs (higher power, more efficient).

I/O availability

Fewer HPC units might mean fewer I/Os than required. While this issue is being addressed by hardware vendors, it is important to consider it during SoC selection.

Supply chain

With chip availability now a challenge, it is critical to understand the impact of maintaining existing supplier relationships over seeking new HPC suppliers promising new capabilities.

Cabling

This is a key consideration in making any design change in the automotive industry. It could be advantageous, for instance, to implement a car door as a zone with its own HPC to avoid additional cabling that separately controls not just the power windows and door locks but also loudspeakers and lights.

Architecture Considerations for Evolution

/ The software considerations in going cloud-ready are:

Modularization

The key consideration is whether legacy software can indeed be modularized or must be lifted and deployed in its entirety in a container.

Overhead

The service-oriented architecture has additional layers of software. While some, like the Wind River® hypervisor, have zero overhead, the RTOS overhead to meet tight timing constraints must be taken into account.

Code availability

Legacy code may not always be available, with older functions only existing as binaries. The lack of future flexibility must be considered against today's additional effort in making the transition.

Effort

Effort and cost: Change requires time, work, and additional expense. With hard deadlines in the background, the overall cost of change is a consideration that dictates its pace

Collaborating to Bring Change

/ Updating the vehicle's E/E architecture is a big ask, but a three-pronged strategy takes much of the effort and pain away.

Open Standards

The cloud-ready vehicle is backed by standards such as the OCI, which has an open governance structure under the Linux Foundation. The Wind River product portfolio is built around open standards that welcome participation from the developer community.

Collaboration

Wind River collaborates with partners in providing tools and helping industries work with a mix of proprietary and open source implementations.

Recent Precedent

Similar architectural change has already been accomplished, most recently by the telecom industry, and offers lessons in revolutionizing product roadmaps. Telcos have evolved from highly specialized proprietary hardware and software implementations to a more commoditized approach, particularly with 5G and the ability there to focus more on value addition.

Conclusion

/ As vehicle architecture continues to evolve, the need for cloudy-ready ECUs will only continue to grow. Wind River provides the tools and technologies that act as a bridge between today's customers and their desired future. Wind River is well positioned to help automotive companies navigate this evolution with our portfolio of products and services that enable intelligent edge computing.

REFERENCES

[1] McKinsey & Co., Outlook on the Automotive Software and Electronics Market Through 2030, January 3, 2023

[2] Data, Cloud-Native Computing Foundation, The State of Cloud-Native Development, May 2022