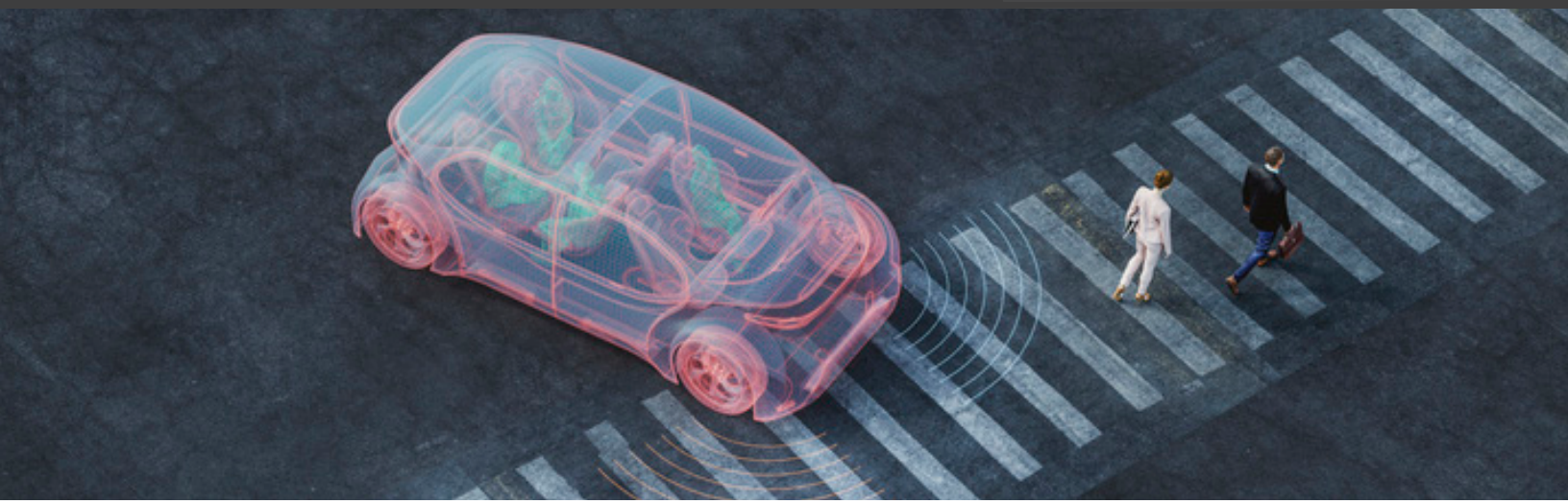


Benefits of Software Containers in Safety-Critical Environments

The transition from hardware-defined to software-defined vehicles is changing everything about the auto industry and beyond. It opens up a world of possibilities for connected, autonomous and electric vehicles, with more innovation than ever before. It also lets OEMs introduce new features quickly, both before and after sale, to meet evolving consumer tastes and needs.

To fulfill its promise, however, the software-defined vehicle requires a hardware architecture that is more optimized and designed for evolution, and a software architecture that is more open and enables a new approach to software development and management. OEMs need the flexibility to modify individual functions rather than issue updates to entire monolithic code bases, and they need a technology that ensures that any changes will not negatively affect adjacent safety-critical software in the vehicle.

Software containerization provides that flexibility. Making containers a reality in the safety-critical automotive environment will require shifts in automotive software architecture, as well as standardization across the industry.

NEW ARCHITECTURES

As software-driven functionality has increased in the automotive world, the compute hardware has evolved. Instead of electronic control units (ECUs) with dedicated hardware and software performing specific functions, high-performance compute platforms have emerged, such as central vehicle controllers (CVCs) and open server platforms (OSPs). CVCs up-integrate body control functions and networking, while OSPs consolidate functions around the specific domains of advanced driver-assistance systems (ADAS) and user experience.

The new approach to hardware saves space and cost, but it also demands a new approach to software. If software is monolithic — as it has been with ECUs — improvements will require full retesting and validation of the entire device it runs in, and any over-the-air (OTA) updates will be much larger than necessary, requiring bandwidth and time.

Key to moving away from the monolithic approach is modularization and abstraction of functions into manageable software blocks, a practice common in the IT world. Controlling a device within the vehicle, for example, is performed by specialized software that handles all signaling to and from the device while presenting a standard, simplified service interface to higher levels of software. This way, the higher-level software does not have to be concerned with the details of how the device is managed, and developers – who are sometimes far removed from the specifics of a vehicle – can focus on just the higher-level logic.

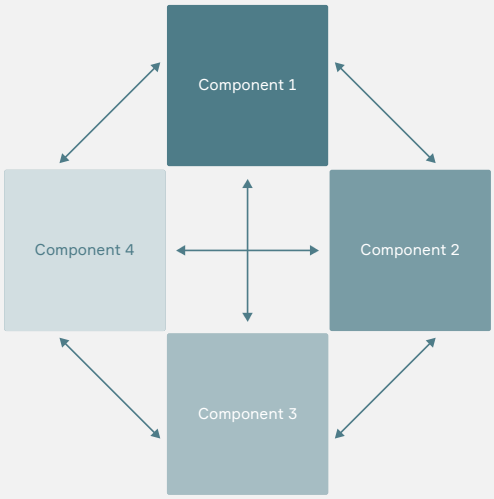
In fact, all vehicle functions can be abstracted and presented as services to other software, creating what is known as a service-based architecture. Software containers fit neatly into this approach.

Leaving Monolithic Architectures Behind

Containerization eliminates interdependencies on common software libraries, allowing each function to be updated independently.

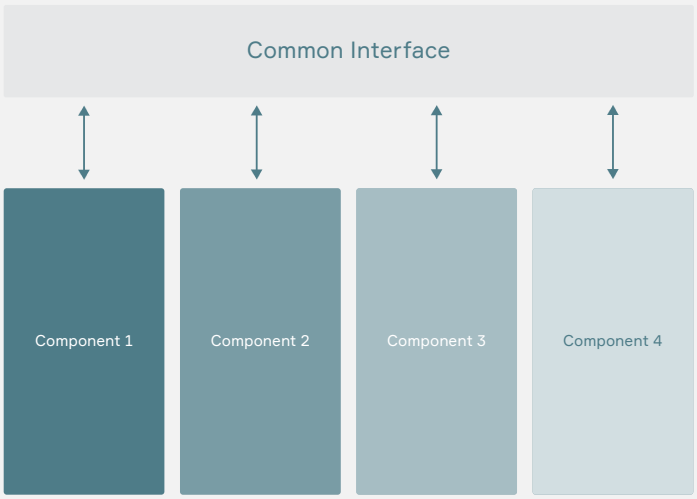
MONOLITHIC

Single, Complex, Intertwined



MICROSERVICES ARCHITECTURE

Standard APIs



Containerization, developed for use in cloud computing, places applications in standard structures that ensure that the dependencies — in the form of service interfaces — among applications are known and controlled. This helps drive stability and consistency in the code base, simplifying isolation of services, and effectively reducing the chances of interfering with other software. It also improves security by keeping attacks that target one application from spreading to others.

By using containers, OEMs and suppliers can adopt modern, agile software methodologies: small teams continually improving individual functions throughout the life of a vehicle with OTA updates, with such changes integrated quickly and automatically. By aligning with open specifications such as that of the Cloud Native Computing Foundation, an OEM and its suppliers can easily update vehicle platforms and even port functions from one platform to another, saving time and money while reducing the risk of errors and systemwide failures.

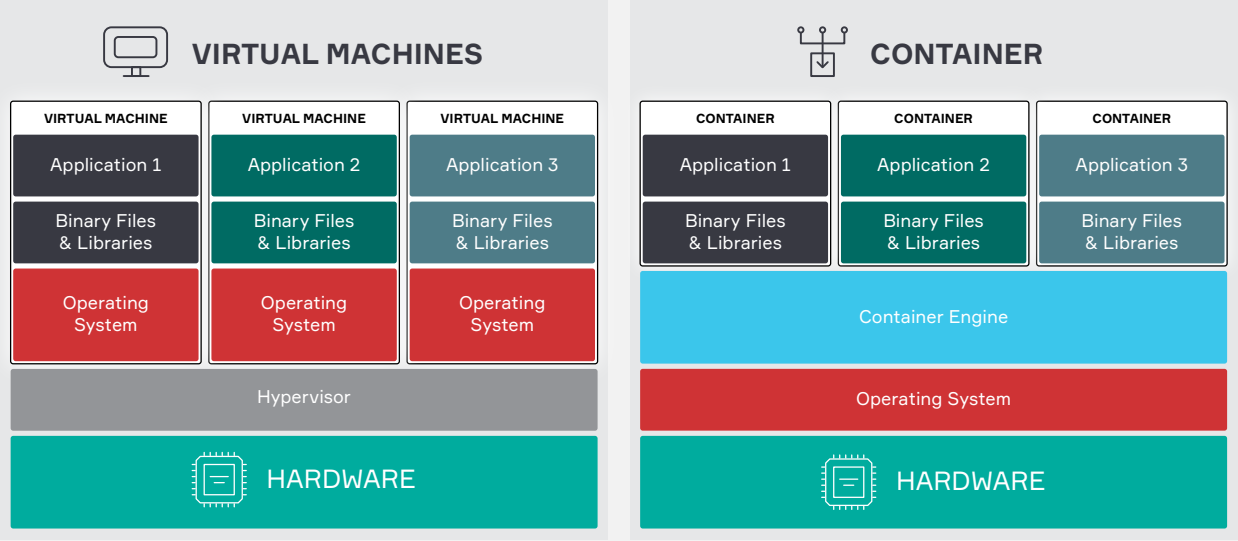
As one piece of the multifaceted transition to software-defined vehicles, containerization is gaining traction across the industry for non-safety-related functions. Because their use in automotive is still new, containers have not yet been certified for use with safety-critical automotive applications such as ADAS, but they are ideally suited to safety features in several ways. Solutions leveraging containers have proven performance in other mission-critical industries, including aerospace, defense, telecom and medical, which helps validate their viability. Key industry players are collaborating to make containers based on the specifications developed by the Open Container Initiative (OCI) that are compliant with industry-standard safety certifications.

CONTAINERS' ROLE IN AUTOMOTIVE COMPUTING

A container is a lightweight packaging format for software. It often includes the application for

Virtual Machines vs. Containers

Hypervisors support virtual machines, each with their own operating system and applications, while containers sit on top of an operating system and container engine, making them lighter.



one function and the specific components that the application requires to run, such as libraries or a programming language runtime. Multiple applications residing in containers can run on one system-on-a-chip or CPU and share processor cycles, memory, storage and networking resources at the operating system level.

Containers were first developed so that cloud-based applications could easily be distributed and scaled up and down. Like virtual machines (VMs), they allow diverse workloads to run concurrently in a shared computing environment. However, VMs are virtualized at the hardware level and incorporate a complete computing environment for the guest application, including an OS. Containers are virtualized at the OS level and are typically smaller. An automotive software architecture might include both VMs and containers: VMs that allow more than one OS on shared hardware, and containers to separate functions under each OS.

Vehicle platforms require three main elements for containerization: a runtime environment that supports containers, a container orchestration system such as Kubernetes, and an OTA network connection for dynamically exchanging containers to and from the cloud in a safe and reliable way. The runtime environment provides standardization among containers and visibility for monitoring the health of each container. A container orchestration system adds another layer for overseeing all containers in a given vehicle.

Containerization turns each function into a standard module that can be added or removed like a Lego® block, making it easier to install, update and manage software. For scalability, the OS can temporarily add more instances of a function at particular times, such as when the vehicle starts up. For portability, a container that performs one function in a comprehensive way can likely be reused in different domains, models and platforms. Containers also ease the integration of legacy software to next-generation vehicles.

In contrast, adding a feature to a traditional automotive code base might require the whole code base to be retested with up to 100,000 miles of driving due to the unpredictable ways in which that feature might interact with others. This approach is growing more costly in terms of time and money as vehicle platforms become more complex.

There are several potential benefits to using containers in automotive environments, including easier OTA updates, increased portability, the ability to adopt modern software methodologies, and coordination across development environments.

OTA updates

The most important way containerization can benefit OEMs is by making OTA updates faster, easier and more frequent. This is becoming essential as more consumers expect — and in many cases are willing to pay for — new features throughout the life of the product. OTA updates offer new ways to increase owner satisfaction and brand loyalty and potentially generate additional revenue.

In traditional, monolithic software environments, after-sale updates tend to be major, infrequent and difficult to distribute over the air. With containerization, they can be smaller and more targeted, both in the number of functions they address and the number of vehicles to which they are distributed. Code verification is standardized and routine, while lower bandwidth requirements mean that most updates can be sent over cellular networks.

Just how much faster and more frequent the updates will be will depend on many factors, not the least of which is how the containers are structured and how lightweight their design is. This is an important area of focus architecturally as developers determine the optimal way to divide applications into smaller pieces, each in its own container.

By ensuring the container structure is optimized, OEMs can more easily perform updates for new features, subscriptions, third-party apps and personalization of individual vehicles. As consumers show interest in new features, dedicated teams can deliver them quickly, which creates an efficient feedback loop.

Portability

Containers also make automotive functions portable across vehicle platforms, models and even vendors. A feature developed for one platform can be deployed throughout an OEM's product lineup with less integration effort. Legacy applications, when placed within standard containers, can easily be integrated into new, containerized vehicle platforms. Portability also prevents vendor lock-in: OEMs can switch to a new supplier more easily, swapping the previous vendor's software for another supplier's containerized application providing identical services.

The portability of containers means the requirements of hardware platforms no longer dictate software development. OEMs can reuse applications as their vehicle platforms evolve from specialized ECUs to domain controllers, to full serverization on central computing engines. Manufacturers can even change the hardware in current models or vehicles already on the road with much lower software update costs.

Modern methodologies

The auto industry has been late to adopt software methodologies that have made other sectors more agile and fast-moving, including DevOps and continuous integration/continuous deployment. OEMs using a traditional waterfall software development process in three-to-five-year development cycles can no longer keep up with technology or consumer tastes.

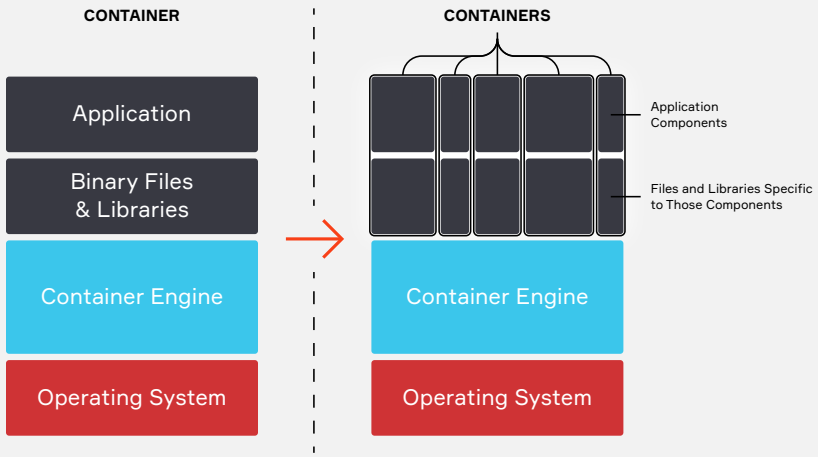
Containers provide the modular architecture needed to quickly accommodate changes in requirements and design. Teams devoted to specific functions can develop and update the software in independent release cycles, owning that code throughout the life cycle of each product or even multiple platforms. Automotive companies that use these methodologies will also have a larger pool of software developers from which to hire.

Coordination

The more that OEMs and their suppliers standardize on the same container specifications, the more they will be able to streamline software rollouts. Most automotive applications go through

Optimized Containers

Containerization and microservices open up new possibilities, and applications themselves don't have to be monolithic. Where it makes sense, they can be divided into smaller modules, each in its own container, making updates more targeted and lightweight.



several environments in the course of simulation, development, testing, verification and production, with separate teams managing each environment. Without containers, the teams need to carefully coordinate any changes to their environment that might affect how those applications run.

Reusing a common container format means that each function has the same context in all environments and that developers can use standard tests to verify whether the code in the container is running correctly. This removes bottlenecks to development in the cloud or any other environment. In addition, once a vehicle is on the road, containerized applications might be able to use the cloud for extra processing power. For example, some automation or infotainment functions could run partly in the cloud, in identical containers.

THE SAFETY CHALLENGE

Traditionally, safety-critical automotive software has been static, isolated in specialized ECUs and limited in size to minimize the overhead of certifying code. In the future, OEMs want to be able to run these functions on shared hardware and update them more frequently.

Co-locating safety-critical functions with other applications creates a need for logical boundaries between applications with different safety and security requirements. Containerization offers a way to implement a modularized architecture via lightweight, easily managed components that can encapsulate safety-critical functions and form those boundaries, so that one container image can be modified without having unpredictable effects on others running in the same environment. This protects safety-critical applications from unexpected failures in noncritical functions that might otherwise spread and violate safety requirements.

The role of VMs

On some vehicle platforms today, such logical boundaries are created by VMs managed by hypervisors with automotive safety certifications. While VMs will continue to play an important role in safety-critical environments, containers present certain advantages. Because more containers than VMs can share a given amount of computing resources, containers can be lighter and require less overhead for verifying safety. In addition, containers are easier to standardize across environments and update over the air.

VMs and containers could be used together in a safety-critical scenario. For example, a VM could run a real-time operating system such as Wind River VxWorks®, and software containers could run on top of that RTOS.

The value in dependability

The well-defined boundaries of containers lend themselves well to applications in situations where it is important to know exactly what to expect.

For example, without containers, developers might not know for sure whether a particular update will overcommit hardware resources until they attempt to load it — and, if it works, they will be hesitant to make any changes for fear of causing it to stop working. In contrast, the defined boundaries of the container mean that developers always know precisely the amount of resources it will use.

Containers also are not persistent — that is, they are loaded from a master image every time the vehicle starts. They do not change over time, and the data they use is stored in a different location. This aspect is attractive from a security perspective because it means that containers can always be counted on to behave the same way, and the services they provide can be considered immutable. It also means that unused applications can be dismissed, freeing up system resources.

A STANDARD IS VITAL

Having a safety-certified runtime environment for containers would ensure that all containers had the resources needed to guarantee safety. No runtime for containers has yet been certified for automotive safety, but efforts are underway.

There is much work to be done to create the processes and documentation required for all of the different development phases.

Standardization can bring the benefits of containerization to more automotive vendors, expanding the market opportunity for next-generation software and the availability of easily integrated applications. In addition, the broad adoption of one container format will create momentum for safety certification.

There is growing industry support for the idea of container standardization, and the OCI container specification offers a proven approach. OCI offers benefits to automotive software development that are similar to what it already provides in cloud computing: Rather than having to adapt applications to each OEM's environment and perform time-consuming integration work, developers could expect to find an OCI-compliant runtime environment on any vehicle platform and write code that could be added to the platform with only minor modifications.

This would go far toward realizing the vision of software-defined vehicles as platforms for applications from a broad ecosystem of suppliers. OCI could play a role like that of the Android OS in smartphones. OEMs could tap into a much larger universe of features, from which customers could choose when purchasing a vehicle or download (or subscribe to) later.

Hurdles to standardization

Containerization is a widely accepted approach to computing but has been used mostly in environments with unconstrained resources. Due to critical safety requirements and tight constraints on space, weight and power consumption, vehicles pose a different set of problems.

Despite this, the challenges of using containers in automotive are less technical than cultural. Several vendors are already working on automotive runtime environments that would support containers and are making efforts to attain safety certifications. To realize the benefits of containerization, OEMs and suppliers also need to change long-standing development practices and shift some resources from integration and testing to innovation and ongoing updates. While all players could eventually benefit, some need to take the initiative to begin the transition.

The first OEM that introduces containerization across its supplier ecosystem might need three or more years to achieve it. But if the industry coalesces around a single standard, such as OCI, others could soon follow in order to derive the same benefits of increased agility, innovation, cost savings, upgradability and time to market. The next step will be for OEMs to implement a container management layer on vehicle platforms, which might take several more years.

SEEDING INDUSTRY EVOLUTION

Aptiv is cooperating with other vendors to accelerate container adoption and standardization across the automotive industry. We demonstrated OTA deployment of OCI-compliant containers to vehicles, using our container orchestration system, at CES 2023. Wind River, the edge software provider Aptiv acquired in 2022, offers an embedded runtime OS that supports OCI containers, and we are working with Wind River to update its automotive safety certification for its newest technology offerings in 2024.

The costs and limitations of using traditional, monolithic software have not become major barriers to OEMs' business models or competitiveness. But as vehicle platforms become steadily more complex and consumers expect safety, comfort and convenience features to evolve more quickly, the pain will increase with each model release. With strong industry cooperation, soon standardized containers that support safety-critical functions will provide an ideal path to more agile development and continuous improvement.

ABOUT THE AUTHORS

Florian Baumann
Senior Director, Software, Advanced Vehicle Architecture, Aptiv

Florian Baumann leads a team focused on developing cutting-edge technologies to solve some of the most demanding problems of the automotive sector. With experience in machine learning, software development, DevOps and cloud architectures, Florian is applying his lifelong obsession with technology to create Aptiv's next-generation in-vehicle software platform, reducing its complexity while providing a seamless developer experience.

Michel Chabroux
Vice President, Product Management, Wind River

Michel Chabroux drives technology and business strategies for Wind River's Intelligent Edge portfolio. He has more than 20 years of industry experience, including roles in technical sales, support, training and product management.

Timm Zimmermann
Senior Director, Software-Defined Vehicle Architecture, Aptiv

Timm Zimmermann leads a team of automotive software architects focused on leveraging Aptiv's Smart Vehicle Architecture™ for software-defined vehicles. Having spent many years in mainframe development and security technologies, Timm is applying his experience toward designing an evolutionary design principle for a next-generation electrical/electronic architecture.

LEARN MORE AT APTIV.COM/MWD →