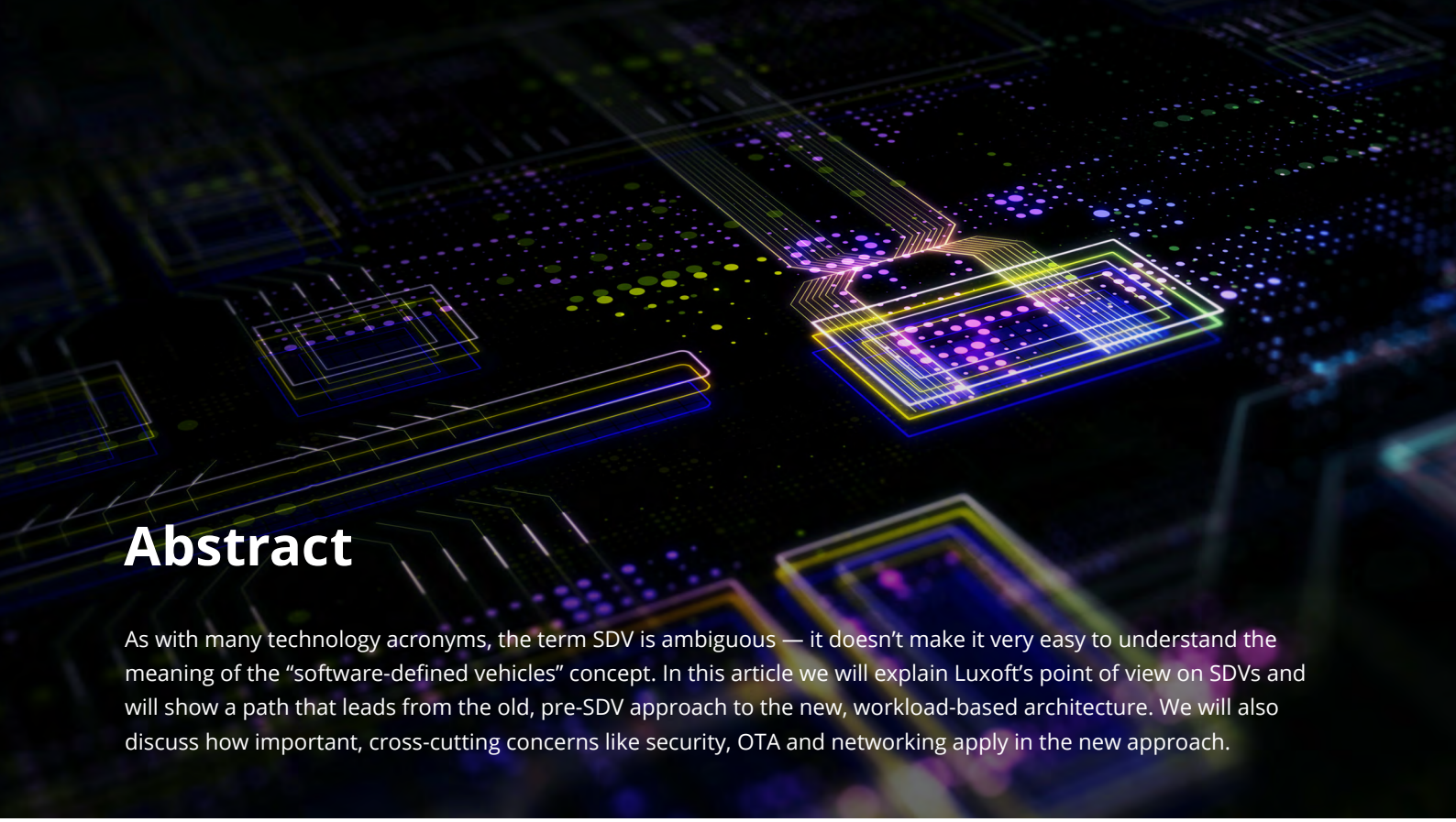
Luxoft
A DXC Technology Company

SDV transition from firmware to workloads

by **Andre Podnozov**, Chief Architect

April 2024



Abstract

As with many technology acronyms, the term SDV is ambiguous — it doesn't make it very easy to understand the meaning of the "software-defined vehicles" concept. In this article we will explain Luxoft's point of view on SDVs and will show a path that leads from the old, pre-SDV approach to the new, workload-based architecture. We will also discuss how important, cross-cutting concerns like security, OTA and networking apply in the new approach.

Historical context leading up to SDVs

Before the concept of "software-defined vehicles" was introduced, vehicles already had lots of software in them. But that software was tightly coupled to specific hardware components called ECUs. At that point, vehicles contained 100+ million lines of computer code spread out among 100+ ECUs, and yet those ECUs were treated as "black boxes" — opaque from a software point of view. In essence, software was being approached with a hardware mindset, which is a limiting factor because software and hardware components are very different in the product lifecycle, maintenance requirements, cost structure, etc.
As the amount of in-vehicle software continued to increase, this inefficient approach to software started to constrain the rate of progress in the automotive industry. At the same time, automotive consumers increasingly expected faster innovation, including new features added throughout vehicle ownership. This tension between the buyers' unwillingness to accept the status quo, and automakers' inability to offer more agile innovation was exacerbated by regulatory pressures to develop electrification (tightening emission standards) and advanced safety assistants (initiatives like Vision Zero). The combination of these three factors created favorable conditions for the SDV revolution.

Our definition of SDVs

Now it becomes clear that it's not correct to say that "an SDV is a vehicle defined by software", but rather we should say, "an SDV is a vehicle that is **not** defined by hardware". Of course, we don't mean mechanical hardware (like wheels, or seats, or body panels, etc.), instead we mean the ECUs — the compute hardware to which software was coupled in the old, non-SDV approach.

But what is the nature of the coupling between software and ECU hardware? It comes from the fact that software is stored in ECUs in the form of firmware, which, as the name suggests, is something between "soft"- and "hard"-ware. Traditionally, firmware was meant to be used for low-level interfacing with hardware devices, such as calibrated reading of sensor values, for example. But since then, a lot of high-level vehicle capabilities have found their way into ECU firmware, and this is exactly what the SDV approach needs to change.

Therefore, here is our full definition of an SDV:

In SDVs, the vehicle capabilities are not defined by ECUs, because the software that implements the capabilities is not burned into ECU firmware.

From firmware → workloads

Even if the software is decoupled from compute, it still needs to be executed somewhere. Where? In pre-SDVs, this question was answered at design time: "Software X (or most likely firmware X) always runs on ECU Y". In SDVs, software X will not be baked into a dedicated ECU as firmware, instead it will be placed as a workload X somewhere on the in-vehicle compute cluster. This placement may:

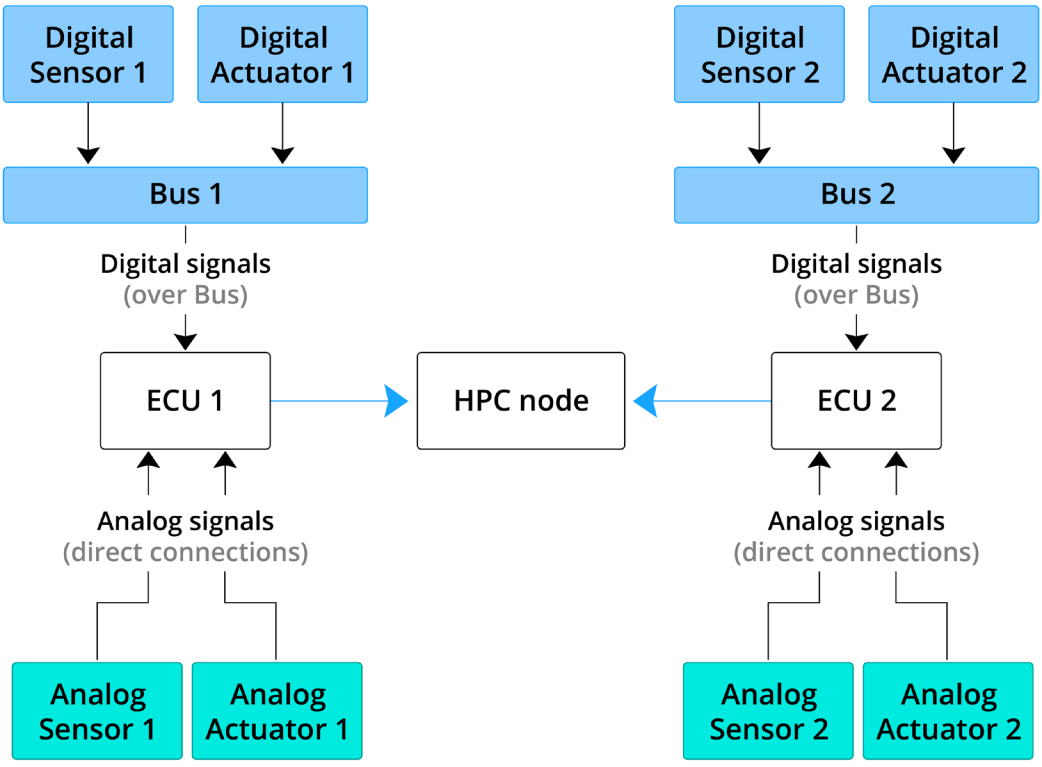
- Be determined at runtime based on compute capabilities required by a given workload (e.g., "execute this workload on any compute node that has AI acceleration capabilities")
- Change during vehicle operation (e.g., "migrate this workload to a different compute node due to hardware failure of current node")
- Be transient (e.g., "this guest workload is started when a passenger boards a taxi, and terminated when the passenger exists"). Such dynamic behavior is necessary to meet the requirements of modern SDVs

If software features are decoupled from ECUs, what do we need ECUs for? In the non-SDV architecture, ECUs have other responsibilities in addition to hosting software features:

- Handling inputs and outputs from "peripherals" like sensor and actuators
- Communicating with components over various in-vehicle networks

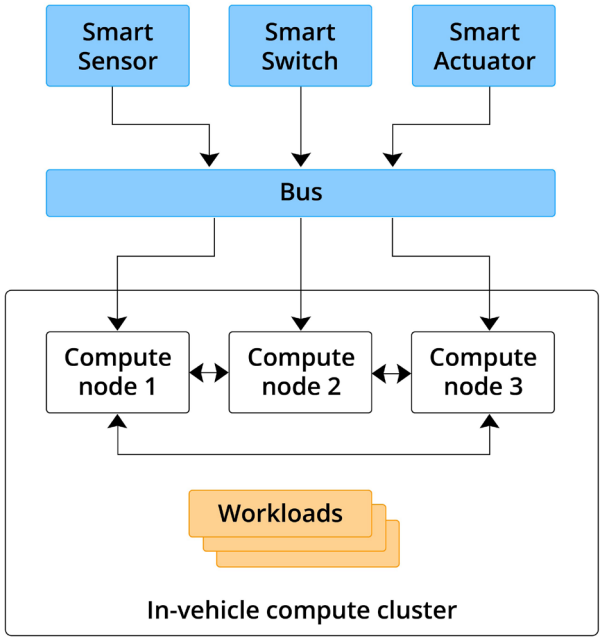
In fact, these two responsibilities are closely related — they bridge the gap between the analog nature of some sensors/actuators, and the digital nature of vehicle networking. Let's look at the diagram below to illustrate this idea.

What about ECUs?



The blue color here represents digital devices and communications, and green is analog devices. Notice that green devices must connect directly to the ECU, as opposed to blue devices that communicate over a bus.

In the SDV world, we can get rid of ECUs completely when all devices become “Smart” and they can perform digital communication, as in the following diagram:



Although, if we are to be 100% technically honest, a blue device is nothing more than a green device with a mini-ECU built into it. But this is exactly how we achieve our original goal: By decomposing a complex entity (big ECU) into orthogonal concerns (software-defined workload vs. signal digitizer + bus I/O) which can later be decoupled in accordance with the SDV architectural principles. Such decomposition is one of the primary instruments for managing the growing complexity of in-vehicle systems.

What is the in-vehicle compute cluster?

The in-vehicle compute cluster consists of several nodes which are shared by various feature workloads. Some nodes may have distinctive capabilities, such as AI acceleration, so that the workloads requiring such capabilities can be placed on the matching nodes. The number of nodes will be at least three (but can be five or even more) — this allows for hardware redundancy and maintaining consensus in the cluster. A five-node cluster may resemble zonal architecture, which can be considered as a half-way step toward a pure workload-based approach.

Cluster nodes should have sufficient resources to host multiple workloads, with room to spare in case they need to accept workloads from a failed neighboring node. Therefore, we are talking about HPC-class nodes, although we can imagine a scenario when ECU-class nodes are also included in the cluster. This would allow for running workloads closer to the data sources. It may also provide a smoother transition path for automakers to start building SDV architectures without requiring a complete hardware revamp.

OTA in SDVs

Because software features should not be baked into ECU firmware in our view of SDVs, it logically follows that OTA should not flash such updates into ECUs. Doing so would be considered an anti-pattern in SDVs. The correct way to update software features would be by using a workload management approach, like our LEAF (Luxoft Edge Acceleration Framework). Examples of workload-oriented OTA updates are containers, modules, ML models, etc.

OTA updates at the workload level are less disruptive than firmware updates because they are:

- Done with no downtime ideally, or in the worst case with very short downtime (<1 minute reboot)
- Transparent to the user in most cases
- Frequent (up to multiple times a day)
- Easier to rollback in case of issues

But ECU firmware can be updated through OTA if the changes are not related to vehicle features. Examples of such OTA updates are:

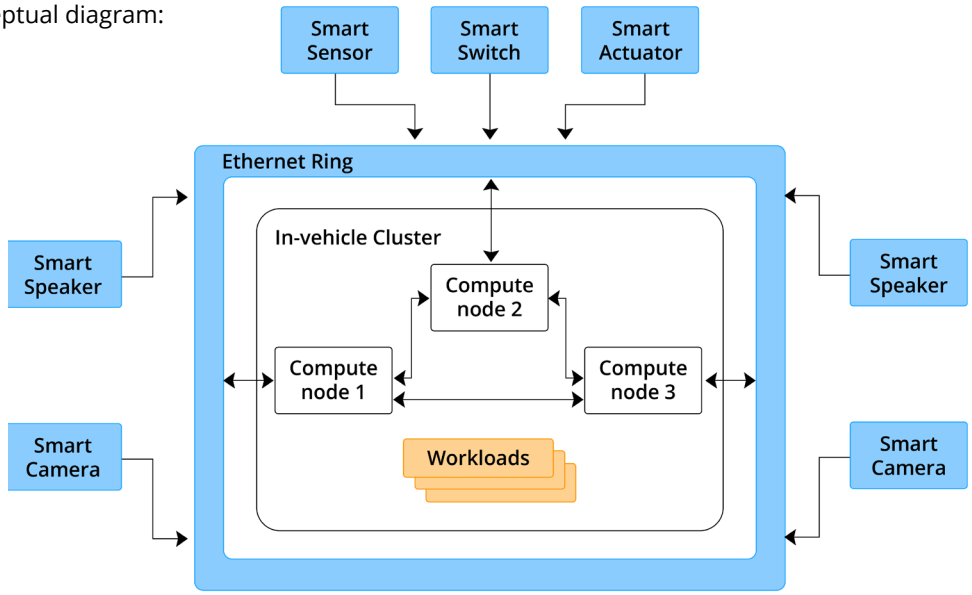
- Updates in mini-ECU functionality of smart devices. As we discussed above, smart devices have mini-ECUs built into them which are responsible for analog<->digital conversion, network communication, authentication and encryption. So, for example, when a network protocol or an encryption algorithm needs to change, it would be done through an OTA update of the smart device firmware. However, these changes should be very rare, and we should consider smart device firmware quasi-unchangeable, after initial bug fixing and stabilization. Also, updates to smart device firmware are much smaller and simpler, because their mini-ECUs have a limited set of responsibilities
- Periodic updates to the cluster runtime (such as Wasm-based) — if the ECU participates in the in-vehicle cluster for hosting workloads. These would go into the firmware. And to re-iterate, workloads themselves would **not** go into the firmware
- Security patches — these should be done as soon as vulnerabilities are identified

To sum up: The majority of OTA updates in SDVs are done at workload level, whereas firmware updates are treated as exceptional cases.

Advantages of bus-based networking

As we are trying to bring good ideas from Big Tech into automotive systems, we should also recognize and take advantage of the unique benefits of automotive architectures. One such benefit is bus-based communication, in comparison to point-to-point networking that Big Tech has settled on. In the bus-based approach, networking hardware (hubs and switches) is not a single point of failure anymore. Bus network architecture also makes it easier to implement redundancy for increased reliability. Most automakers have adopted Automotive Ethernet for high-speed communication in their vehicles. For example, Tesla used the term “Etherloop” when describing their latest Cybertruck networking, and they explicitly mentioned redundancy as one of the benefits.

Here is a conceptual diagram:



Security considerations for SDVs

Pre-SDV architectures treated the in-vehicle environment as a trusted system. However, the experience of building connected products in other domains has taught us to apply defensive security principles like “Zero Trust”, “Expect Compromise”, “Defense in depth”, etc. to the in-vehicle environment. The automotive industry has dedicated a lot of attention to cybersecurity of in-vehicle systems. However, new attacks continue to surface, such as the recent compromise of secure communication in Time-Triggered Ethernet.

Therefore, SDV architecture must continue to prioritize security of in-vehicle systems. We can borrow some of the security best practices from the IoT toolset, such as:

- Certificate-based authentication
- Individualized device identities
- Encrypted communication
- Hardware root of trust

Which means that every smart device that connects to the in-vehicle network must support the above techniques. It may seem computationally expensive to encrypt all communications on a microcontroller-class chip typical of a mini-ECU inside a smart sensor. However, new cryptographic algorithms are being developed that are optimized for low-compute devices.

Luxoft contribution

Luxoft has been putting continued efforts into the development of SDV architectures based on the workload approach. Here are a couple of recent examples:

- SDV Actors whitepaper describes a workload-based architecture implemented using the Actor model of computation
- Luxoft Edge Acceleration Framework post describes an SDV environment that uses container workloads

Stay tuned for future publications where we will describe our development of additional types of in-vehicle workloads, such as ML models.

Conclusion

The view on SDVs described here helps automakers manage the growing complexity of in-vehicle software — it enables the highly flexible dynamic behavior that will be required of tomorrow’s autonomous vehicles. At the same time, it allows agile delivery of vehicle features expected by today’s customers.

To discuss your needs regarding SDV transformation and adopting the workload approach, **get in touch with one of our experts through the contact page.**

About **the author**

Andre Podnozov
Chief Architect, Connected Mobility
apodnozov@dxc.com

As a seasoned technology leader at Luxoft, Andre has a wealth of expertise regarding the future of mobility. In his role as Chief Architect of Connected Mobility, he's shaping the landscape of software-defined vehicles, leveraging the latest advancements in IoT, AI/ML, Digital Twins and other exponential technologies. By bridging the gap between Cloud, Edge and In-vehicle systems, Andre is revolutionizing the way we think about transportation.

About Luxoft

Luxoft, a DXC Technology Company, is a trusted partner in global digital transformation and a leader in delivering competitive advantage in the software-defined world. We engineer and deliver innovative services and products that shape the future of industries by leveraging our extensive partnership network and deep industry-specific expertise.

For more information, please visit **luxoft.com**