Luxoft
A DXC Technology Company

The agility challenge for software-defined vehicles

Abstract

In this white paper we outline the need for a more agile way of developing software-defined vehicles (SDVs). We consider the possibility of borrowing established methodologies from the software industry and applying them to the automotive industry. We explore the opportunities this holds for automotive, and discuss the challenges that prevent direct transplanting of certain techniques and approaches, such as Microservices and DevOps, from the cloud to the in-vehicle environment. We conclude by paving the way for a potential solution to the agility challenge discussed in this paper.

A shift in expectation

Consumer preferences in the automotive industry are changing. Traditionally, vehicles were viewed through the lens of their mechanical characteristics: Which car has more horsepower, a louder sound system, larger wheel diameter, etc. However, modern automotive consumers increasingly prioritize intelligent vehicle features that are enabled by advancements in software rather than hardware. For example, customers want intelligent driving assistance features to help them arrive at the destination more relaxed, regardless of the 0-60 times. Both busy workers and busy families want to keep up with all sides of their work/life balance while on the go, and simply having hands-free calling in the car doesn't cut it anymore. People want cars to be a dependable asset that never surprises them with an unexpected breakdown. Moreover, users want cars that become more capable as time goes by, with new features and improvements added through a seamless Over-the-Air (OTA) update, rather than cars that gradually become obsolete with each passing day. In these examples, consumers identify the car's value proposition mainly in terms of intelligent features that can only be made possible by advanced software. In other words, the shifting consumer perspective is heavily focused on SDVs.

In order to meet customers' expectations, automakers are reinventing themselves to become more agile in delivering intelligent features throughout the life of the vehicle. This process of reinventing is similar to the digital transformations we've seen in other industries, so we can extract useful learnings from success stories in those domains. Also, can we adopt some of the proven tools and methodologies for automotive purposes? Since we're talking about software-enabled vehicle capabilities, the obvious place to look for such inspiration would be the software industry.

Of course, the automotive industry is not a newbie when it comes to software development, quite the contrary — sophistication of automotive in-vehicle software exceeds some of the most well-known examples from the software industry itself. Modern cars can have over 100 million lines of software code, which is more than Microsoft Windows or Office products (and more than almost everything in this comparison: **visualcapitalist.com/millions-lines-of-code/**).

However, in addition to being more complex, automotive software has much stricter quality and safety requirements, which is completely understandable given the risks involved. Therefore, the software industry's favorite motto '**move fast and break things**' (made famous by Facebook), would be inconceivable in automotive.

How the agile challenge was solved on the cloud

The need to be more agile is not unique to the automotive industry — the software industry has been actively solving the same challenge for more than 2 decades. By now, it has developed a highly effective set of approaches for dealing with this challenge — the combination of Microservice and DevOps. Let's look at why this combination is so effective for cloud-based services, and then try to imagine how this could work for in-vehicle software.

Microservices

Microservices were a logical step in the evolution of service-oriented architectures; they allowed larger development teams to effectively collaborate on more complex projects. This meant new features could be continuously delivered straight into the cloud services, sometimes as often as multiple times per day. Contrast that to the era of boxed software when customers had to wait several years to get new features.

Microservices also enabled more efficient ways of operating cloud services at higher scale. At the same time, the cost of innovation decreased rapidly, in accordance with the 'economies of scale' paradigm (often into Zero Marginal Cost territory as evidenced by the 'forever free' services such as Gmail). All told, Microservice architectures, backed by modern cloud technologies, allowed companies to stay relevant in the fast-paced software industry, even in the face of quickly changing user preferences and viral trends.

DevOps

It's well known that DevOps methodology and Microservice architecture go nicely together. But for internet-facing services that were built on Microservice architecture, DevOps was not a choice — it was a necessity. The newfound agility of development that was unlocked by Microservices was a double-edged sword — yes, it resulted in faster updates, but it also brought greater risk of destabilization from all the changes being constantly deployed in all parts of the Microservice application. This is akin to the folk wisdom, "A 4-wheel-drive car will only help you get stuck farther down into the swamp, thus complicating recovery efforts".

DevOps became the go-to recovery mechanism for any Microservice system failures, either from a bad code update that slipped past QA into production, or from sudden spikes in popularity of a website resulting in a flood of internet traffic (or for any other reason). The complexity of Microservice systems quickly grew to the point that any issues with the system required immediate investigation by the Devs (developers) who implemented it, as opposed to Ops — System Operations Engineers (as was previously the case). This is how the original DevOps, (aka '24/7 on-call sleeping-with-a-pager do-not-leave-home-without-laptop') practice was born.

Let’s try the same approach in-vehicle

Since automotive also has the agile challenge, why not try to solve it with in-vehicle Microservices? Microservices would bring significant agility benefits to the SDV, but unfortunately, they would also bring significant risks that we can’t afford. Namely, Microservices can solve the agility challenge on the cloud only when combined with DevOps, but Cloud DevOps methodology cannot be transplanted into the vehicle. There are several problems:

Constrained connectivity

Cloud systems are always operated under the assumption of full connectivity — loss of connectivity would be considered a high-priority incident. Connected vehicles, on the other hand, are designed to operate normally with intermittent connectivity, or in conditions of constrained bandwidth. This would make it impossible to rely on a remote DevOps engineer’s intervention every time a car experienced a software issue. And if a car was transitioned to Microservice architecture, software issues would be plentiful.

Size of the fleet

Let’s assume that in time, every car will have perfect, continuous unconstrained connectivity — even in this case, we wouldn’t be able to operate a fleet of connected vehicles the same way as a cloud system, due to the very large size of vehicle fleet. Let’s try to estimate the size of a connected vehicle fleet for any of the top three global manufacturers (which produce about 10 million vehicles each per year). Automakers have already had connected vehicle technology for at least 5 years, so we can estimate the size of the fleet at 50 million vehicles for each top manufacturer. Now let’s remember that a modern vehicle contains over 100 small computers called ECUs, connected into a distributed network inside the vehicle. Therefore, the total number of hardware devices that need to be monitored in our example fleet exceeds 5 billion. This number is 3 orders of magnitude greater than the largest cloud server fleet of the top three cloud providers. Microsoft has about 100,000 engineers, most of them working on Azure cloud. How many engineers would it take to operate a fleet 1000 times larger? Half the adult population of the USA? The numbers are simply not feasible.

Automotive-grade quality

Even if we could (hypothetically) find enough engineers to babysit each individual vehicle, the high standards of automotive quality would never permit the number of software bugs/failures/outages that are accepted in various internet services. This goes back to the 'move fast and break things' philosophy that has now spread to the farthest corners of the software industry.

Non-elastic compute environment

Many approaches of Cloud DevOps rely on the unlimited elasticity of compute resources in the cloud. For example, if a cloud host experiences failure, you can just move to a fresh one. However, the in-vehicle compute environment is pretty much all accounted for; there are no reserves waiting. On the bright side though, unlike a cloud system with its unbounded characteristics, the in-vehicle environment and its behavior can be fully understood (at least in theory).

Different security model

Security is the number one concern in connected systems, especially in high-consequence scenarios like automotive. At the same time, security is the area where the gap between the cloud world and the in-vehicle world is the most evident. The traditional model of cybersecurity relies on the fact that our services run in physically secured datacenters — be it public or private cloud, or even edge datacenters. But cars are operated in environments where unfriendly actors can potentially have physical access to the vehicle's computing hardware, and according to a well-known rule of IoT security, a computer system cannot be protected against compromise if an attacker can gain physical access to the system's hardware. Whereas the cloud DevOps process has a goal of 100% protection against cyberattacks, a fleet of SDVs must be operated under 'assume breach' philosophy in which any vehicle can be compromised — it's only a question of how much an attacker wants to invest in the attack. Then it becomes a matter of raising the protection bar high enough to deter all but the most determined attackers (such as nation-state backed entities). In any event, a compromise of any single vehicle should not endanger the security of the connected system as a whole. Fortunately, there is a good body of knowledge accumulated in the technology industry on securing IoT systems — this can be taken as a foundation and adapted to automotive specifics.

The solution could be just around the corner

Hopes are high today that the SDV paradigm can unlock a new era of automotive progress rich with transformational outcomes. However, the SDV success story can only be written if the automotive industry can solve the agility challenge. Fortunately, this challenge has already been solved in other domains using methodologies like Microservices and DevOps. Can the automotive industry take a page out of the software industry's book and apply these methodologies to SDVs? We believe so. The key could be open standards and collaborations such as **our work with SOAFEE**, and our endeavor to advance an automotive-grade open technology platform for the automotive industry with **the Eclipse Foundation**. Stay tuned for our next paper where we'll introduce a new concept that provides a positive outlook to the challenge.

About **the authors**

Andre Podnozov
Chief Architect
apodnozov@dxc.com
linkedin.com/in/andrepodnozov/

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 by 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.

Andreas Lindenthal
Automotive Consultant, AD Architecture
andreas.lindenthal@dxc.com

Andreas is an Automotive Consultant at Luxoft. His key areas of interest are processes, methods and tools with focus on digital transformation. His background ranges from automotive hardware architectures, real time embedded operating systems, automotive sensors and infotainment systems development.

About Luxoft

Luxoft, a DXC Technology Company delivers digital advantage for software-driven organizations, leveraging domain knowledge and software engineering capabilities. We use our industry-specific expertise and extensive partnership network to engineer innovative products and services that generate value and shape the future of industries.

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