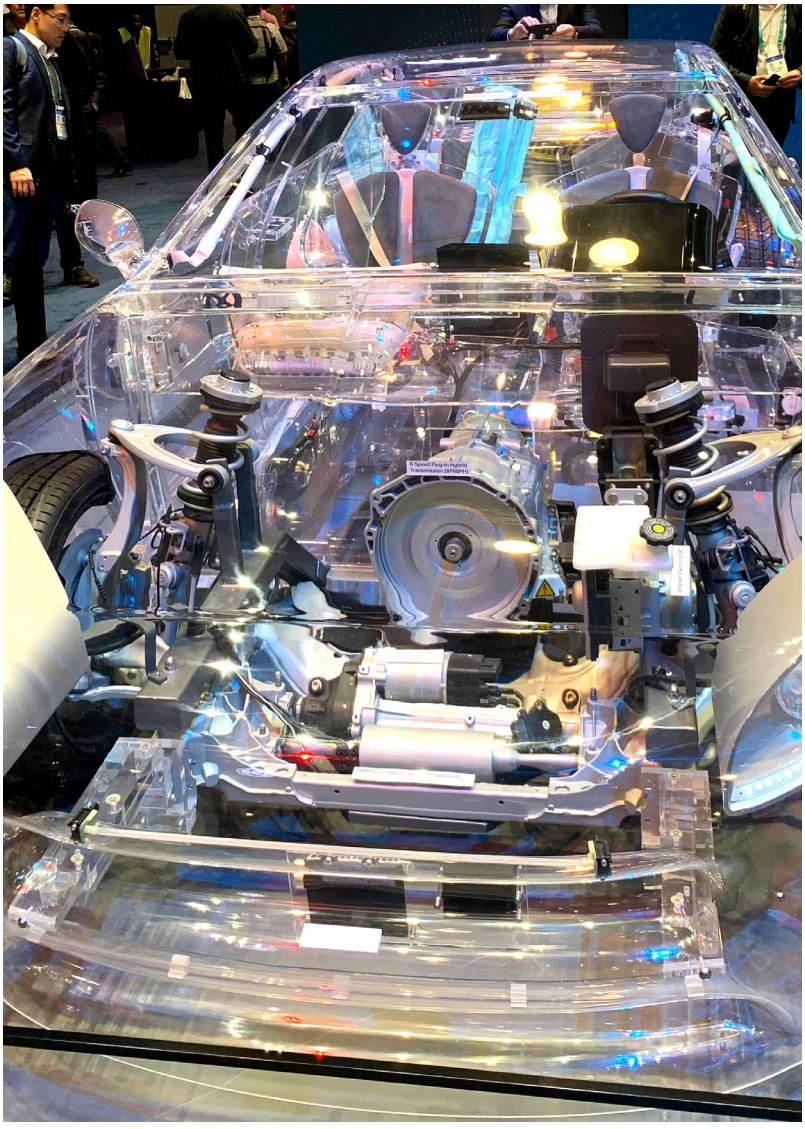
Photo by Kearney alumni



Reality check: five approaches to the software-defined vehicle from the past decade

ETAS | BinaryCore | KEARNEY

Forward-thinking automakers can thrive in the SDV market by not only learning from the past but also setting their sights on the future.

The software-defined vehicle (SDV) has been around for the past 10 years. And while software natives and electric vehicle players may be farther along than others, now is a good time for all original equipment manufacturers (OEMs) to see what they can learn with a little hindsight.

In this article, we explore five common approaches to the SDV, examine the successes and challenges, and look at the current state of development and implementation. The insights offered here come through a collaboration between the global consulting firm Kearney and ETAS, a leading automotive software supplier. This article provides comprehensive solutions for organizations looking to succeed in the SDV market, from strategy to development and testing systems across vehicle domains.

Now is a good time for all original equipment manufacturers (OEMs) to see what they can learn with a little hindsight.

Approach 1: decoupling hardware and software

Why did this approach come about?

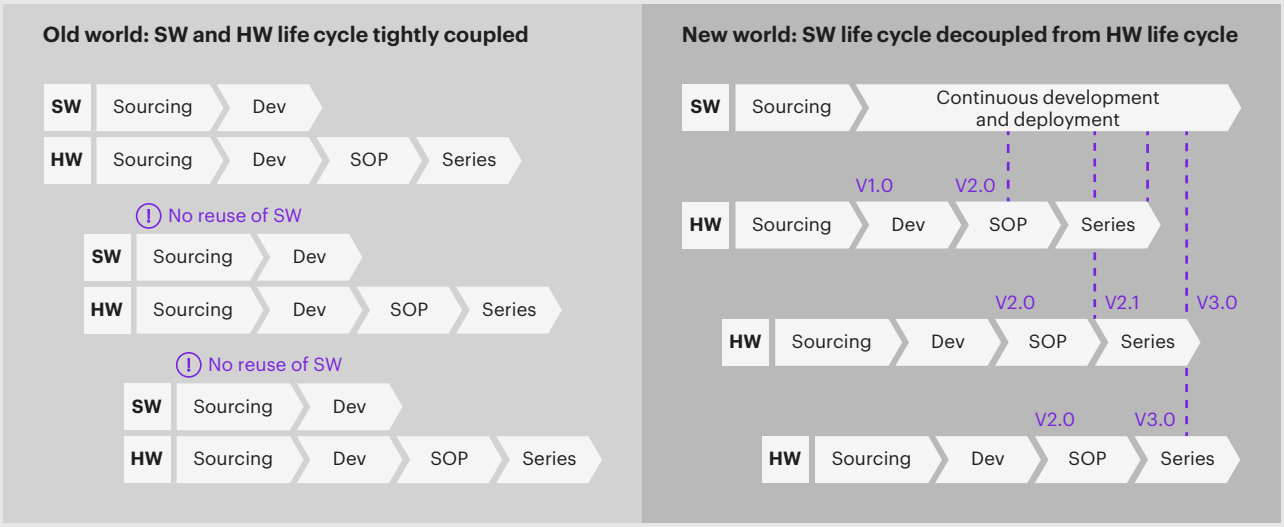
For decades, software was an addition to hardware. Traditionally, OEMs single-sourced electronic control units (ECUs) with the software integrated. Many of these ECUs were hard-wired in decentralized electrical/electronic architecture. Features were released on old, disconnected vehicle generations, and the lack of interoperability and high complexity led to long development cycles and higher operational expenditures. In this world of planning and tightly coupled software and hardware development life cycles, the OEM would source hardware and software for the new vehicle as a bundle. All the required software elements would be developed up to the start of production. As a result, integration became incredibly difficult because the vehicle's functionality would depend on several suppliers.

When challenger OEMs arrived in the SDV market, they caused a paradigm shift by focusing on innovation through software instead of hardware. Their decoupled software and hardware approach paved the way for continuous development and deployment. OEMs used to plan for at least three years to develop software in parallel with hardware. Now, the process was faster and more iterative up to the start of production and beyond, and it included the entire vehicle life cycle. Software modules from multiple suppliers were seamlessly integrated, and over-the-air updates enabled ongoing maintenance and created new revenue streams from innovating customer-facing features. As the industry progresses, the sourcing of hardware and software functions will be decoupled in the sense of not only two separate offers but also the life cycle. Functions will be continuously developed and added to a series of vehicles, not just for one specific vehicle at a time (see figure 1 on page 3).

As the industry progresses, the sourcing of hardware and software functions will be decoupled.

Figure 1
Approach 1: decouple hardware and software

The next generation of E/E architectures enables continuous development and deployment of software updates over the air.



Notes: SW is software. HW is hardware. SOP is start of production.
Source: Kearney analysis

What worked?

This shift caused leaders from across organizational functions to recognize the need to decouple software from hardware as much as possible. A series of M&A transactions helped build the required software capabilities or incorporate products to integrate with the platform. The portability of functions moved into the market with tier 1 suppliers producing software in modules. Integration of these modules from one organization into an ECU from another became standard. Software-native OEMs stuck to agile principles of developing an MVP platform and continuously enhancing the feature stack to manage complexity.

What didn't work?

Many traditional OEMs took a big-bang approach, forcing the full software feature set from day one. Starting with an MVP is essential for agile development, but in this instance, organizations tried to leapfrog it by throwing money at the work and developing a full-function scope. Because it's normal to hit a software boundary when first developing hardware, it became common to see hardware under specification because of risk-averse finance departments. Handling legacy topics is also always an issue. The intention is to start fresh, but over time, legacy features begin to creep in. And once one legacy feature is in play, it could be connected to many others—starting a chain of interdependency. Lastly, many OEMs tried to do everything at once instead of focusing on a certain domain. Few effectively established the fundamentals while focusing on a limited set of differentiating features to pursue a manageable development scope.

Approach 2: establishing a separate software unit

Why did this approach come about?

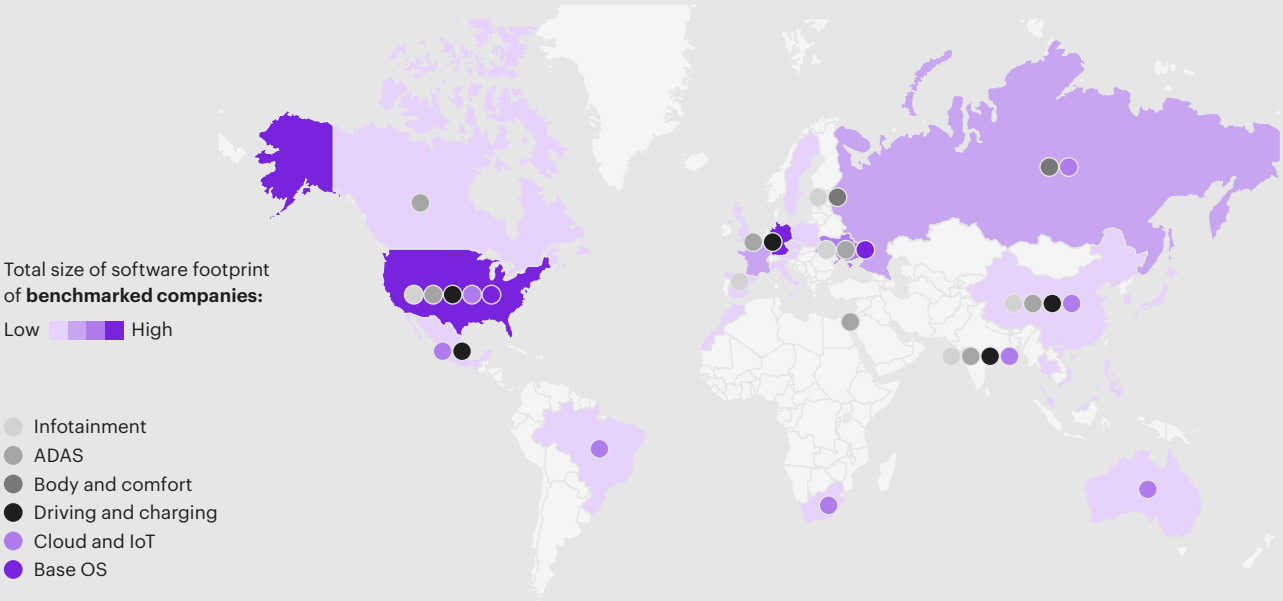
In a word, talent. Successful software development relies on creating a space for people with the right mindset and skills to come together and develop free from legacy. Creating such a culture is essential, as reflected in the ongoing war for talent with OEMs competing against the biggest names in tech. Organizations that don't have a favorable software culture simply won't be considered by the best and most in-demand talent, who are drawn by not only bigger salaries, but also better working conditions and more exciting projects.

Traditional OEMs historically grew their R&D departments with a hardware-first approach, meaning teams were great at producing cars in a waterfall approach with solid planning and execution in a two- or three-month cycle. However, this also meant they lacked the skills to rethink vehicle architectures for the SDV and struggled with a lack of specialist capabilities in areas such as telematics, cloud back end, and artificial intelligence. Software requires a new way of thinking. Hardware focuses on specification because building costs money, but software is the reverse: you can't specify because it's impossible to predict every dependency and state, so the focus is on integration. Software-first organizations can implement more agile methods, measure the result, and make refinements at faster intervals (see figure 2 on page 5).

Software-first organizations can implement more agile methods, measure the result, and make refinements at faster intervals.

Figure 2
Approach 2: separate software unit

The global automotive software footprint showing the total size and focus domains of countries



Notes: ADAS is advanced driver assistance system. IoT is Internet of Things. OS is operating system.
Source: Kearney Automotive Software Benchmark Index

What worked?

This approach allowed organizations to grow and develop their software culture with limited interference. Inside the unit, it enabled more cross-functional thinking. OEMs stopped focusing too much on specific domains with sporadic communication between separate teams, causing management overhead. Instead, they could build new features by drawing from the right pool of talent on day one. Also from the outset, the focus on integration ensured the frequent and smooth synchronization of hardware and software.

What didn't work?

Many OEMs structured software development in a way that focused exclusively on their vehicle lines and launches, and they put the actual software second. Too often, existing R&D and corporate structures were the template for the software unit, and hardware talent shifted to work on software. Far from growing a software culture, the lingering presence of the parent organization slowed decision-making and hampered innovation. Teams also grew too big and too fast with a low make–share ratio. Some even numbered around 6,000 people. In short, the "now we change everything approach" was too ambitious for a one-step transformation and led to rejection by the surrounding organization.

Approach 3: developing the full software stack

Why did this approach come about?

Many OEMs have a desire to develop the full software stack in terms of domains, such as advanced driver assistance systems (ADAS) and in-vehicle infotainment, as well as the ECU tech stack (the operating system, middleware, and applications). Over the past few years, the race has been on to increase the make ratio across everything from the hardware to the firmware, middleware, and applications. Organizations are attracted by the idea of reducing the dependency and effort spent on integrating software suppliers and gaining complete strategic control with the in-vehicle infotainment domain, drawing many similarities to the smartphone ecosystem. To put the scale of this ambition in perspective, even the leading SDV players develop only the application and middleware layers while adapting Linux firmware.

What worked?

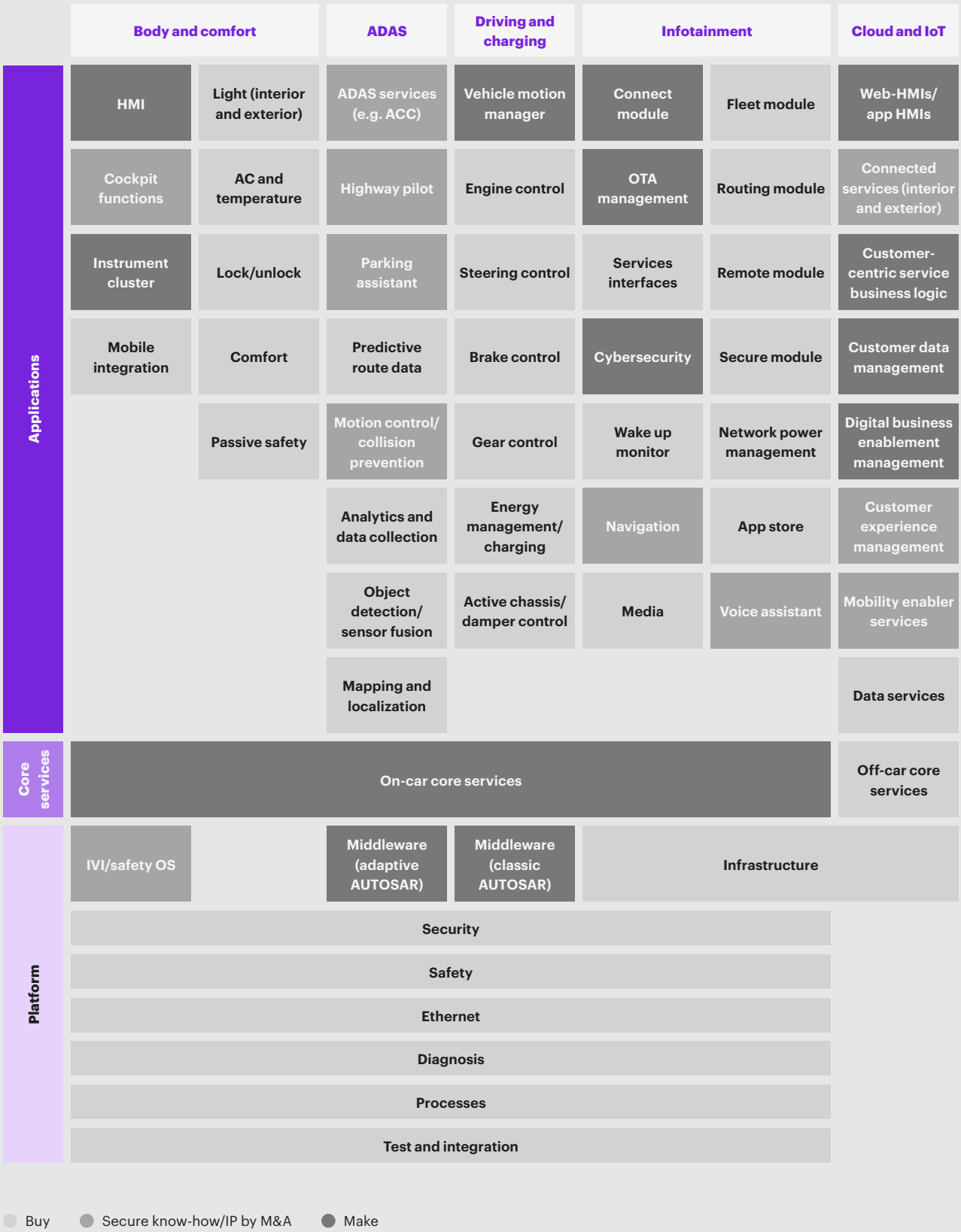
A positive by-product of this approach was a new focus on the in-house development of customer-facing and differentiating software components. OEMs began to prioritize areas where the customer's good impression of the user experience would benefit the manufacturer, such as a touchscreen navigation system or infotainment features. Beneficial partnerships and open-source communities emerged for platform development, such as Automotive Grade Linux, Android Automotive, and the newly founded Eclipse SDV. In combination with the platform and middleware open-source communities, multiple open standardization efforts are focusing on other challenges, such as the Connected Vehicle Systems Alliance (COVESA) for data and service and the Scalable Open Architecture for Embedded Edge (SOAFEE) project for defining a cloud-native architecture and behavior all the way down to the chipset level (see figure 3 on page 7).

What didn't work?

There were issues right from the definition phase, with various interpretations of what developing the full stack meant. OEM-specific *.OS projects ranged from pure application to middleware and base OS on up to the full ecosystems around the car. The overall stretch in resources, time, budget, and capabilities was overwhelming, with OEMs trying to support the demands of brands and vehicle platforms that might have different ways of categorizing the same data point or service API. This is the exact opposite of the desired uniform model in which a single definition or catalog streamlines data across brands and increases the reuse of code that interacts with vehicle signals and services.

Figure 3

Approach 3: Developing the full software stack



Notes: OEM is original equipment manufacturer. COP is conformity of production. ADAS is advanced driver assistance system. IoT is Internet of Things. HMI is human machine interface. AC is air conditioning. ACC is adaptive cruise control. OTA is over the air. OS is operating system. IP is intellectual property.

Source: Kearney analysis

Approach 4: agile as the software holy grail

Why did this approach come about?

We are now at the tail end of a period in SDV history that saw everyone using agile delivery methods for everything. In much the same way, OEMs were using a waterfall approach in the form of V-model cycles for all hardware development. With the prioritization on software development and its lack of need for physical building, the faster innovation cycles and flexibility were more appropriately matched to agile delivery. OEMs saw the benefits of agile as the proven method of the high-tech software industry and perceived it to be the best, often taking a blanket approach (see figure 4 on page 9).

What worked?

Adapting agile for the right domains, such as application and back-end development, can have a big impact. When functional teams are responsible for the end-to-end software value chain from design to development and integration across domains, they are more likely to adopt the right mindset across teams. The decision of what delivery model to use depends on the specific set of circumstances, such as when opting for a lean approach rather than a scaled agile framework (SAFe). In a two-speed model, with agile used for cloud and in-vehicle features and the traditional V-model used for the base operating system, electronics and mechanics can sustain velocity throughout feature and platform development.

What didn't work?

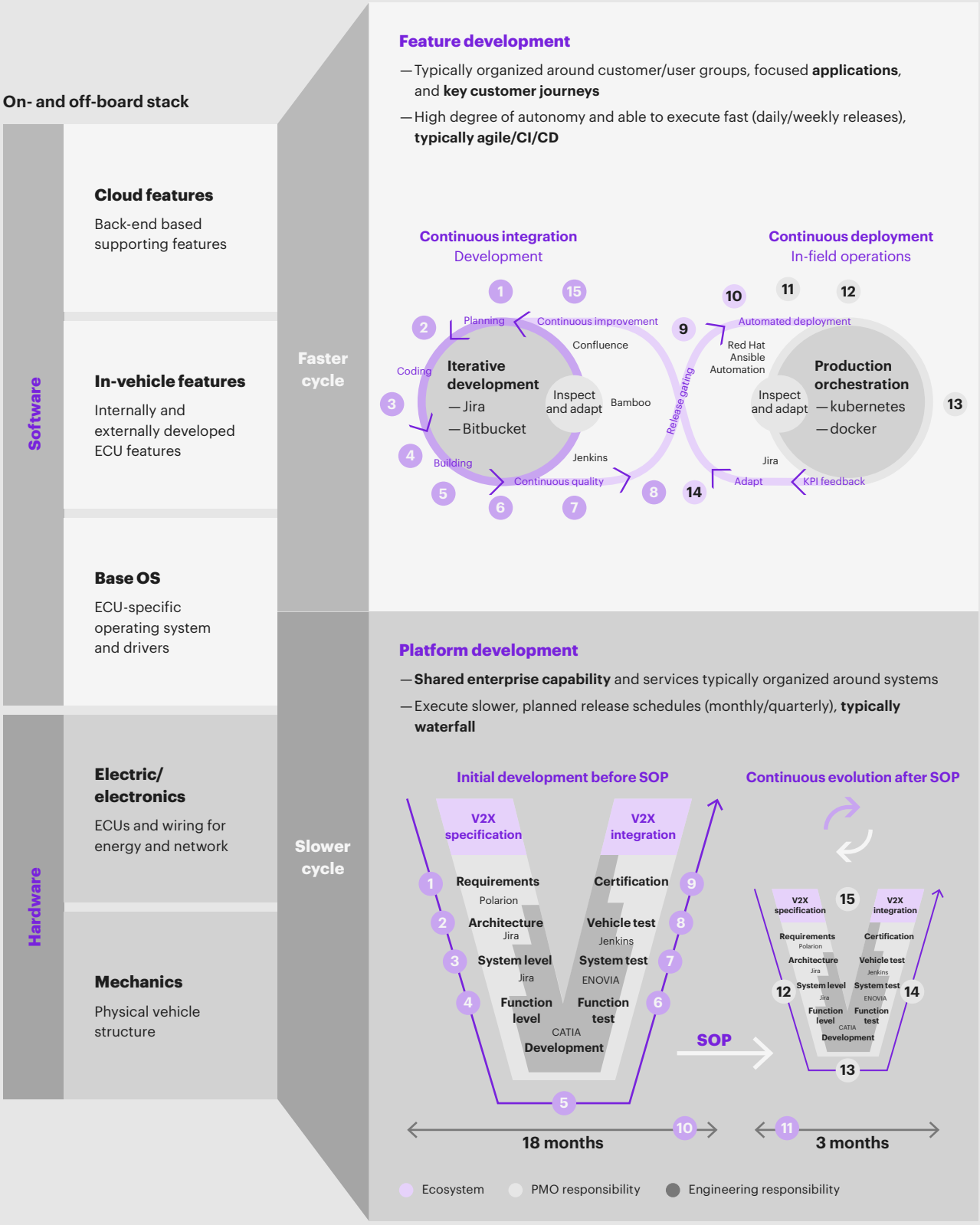
It doesn't work when agile is used for everything, all the way from applications and platforms to business functions such as purchasing and marketing. Short-term reskilling bootcamps reinforce the mistaken view that agile can simply be adopted without a new long-term mindset. There have been too many accounts of system architects wholly responsible for the design without the time to oversee and guide the delivery. A SAFe approach can be the worst of both worlds, with agile acting as the shiny cover hiding the core of traditional waterfall management. There's also the issue of finding the right synchronization between fast-cycling software features, mid-pace cycling of the base operating system, and the slow-cycling of hardware.

Figure 4

Approach 4: agile as holy grail

The digital and fast product creation must provide a "two-speed" delivery model to sustain velocity.

Simplified



Notes: ECU is electronic control unit. OS is operating system. CI/CD is continuous integration/continuous development. SOP is start of production. PMO is project management office.

Source: Kearney analysis

Approach 5: tool landscape transformation

Why did this approach come about?

At its core, software and hardware development used to be encapsulated with each department able to build its own toolchain. The result was an array of tools being used by various departments for management, design, testing, and operations. OEMs managed to work with as many as 1,000 different tools. With the SDV, this encapsulation no longer works because of the mass of interconnected software functions—driving the need for one integrated toolchain that could cater to the needs of an agile environment, continuous software development, and integration (see figure 5 on page 11).

What worked?

On the one hand, this approach brought about an all-new level of focus on building single end-to-end automated toolchains. Some OEMs were able to reduce their set of core tools with a uniform configuration; SDV leaders only use around 20 core tools. It also helped integrate suppliers and external companies, aiding their collaboration on a central database and tools.

What didn't work?

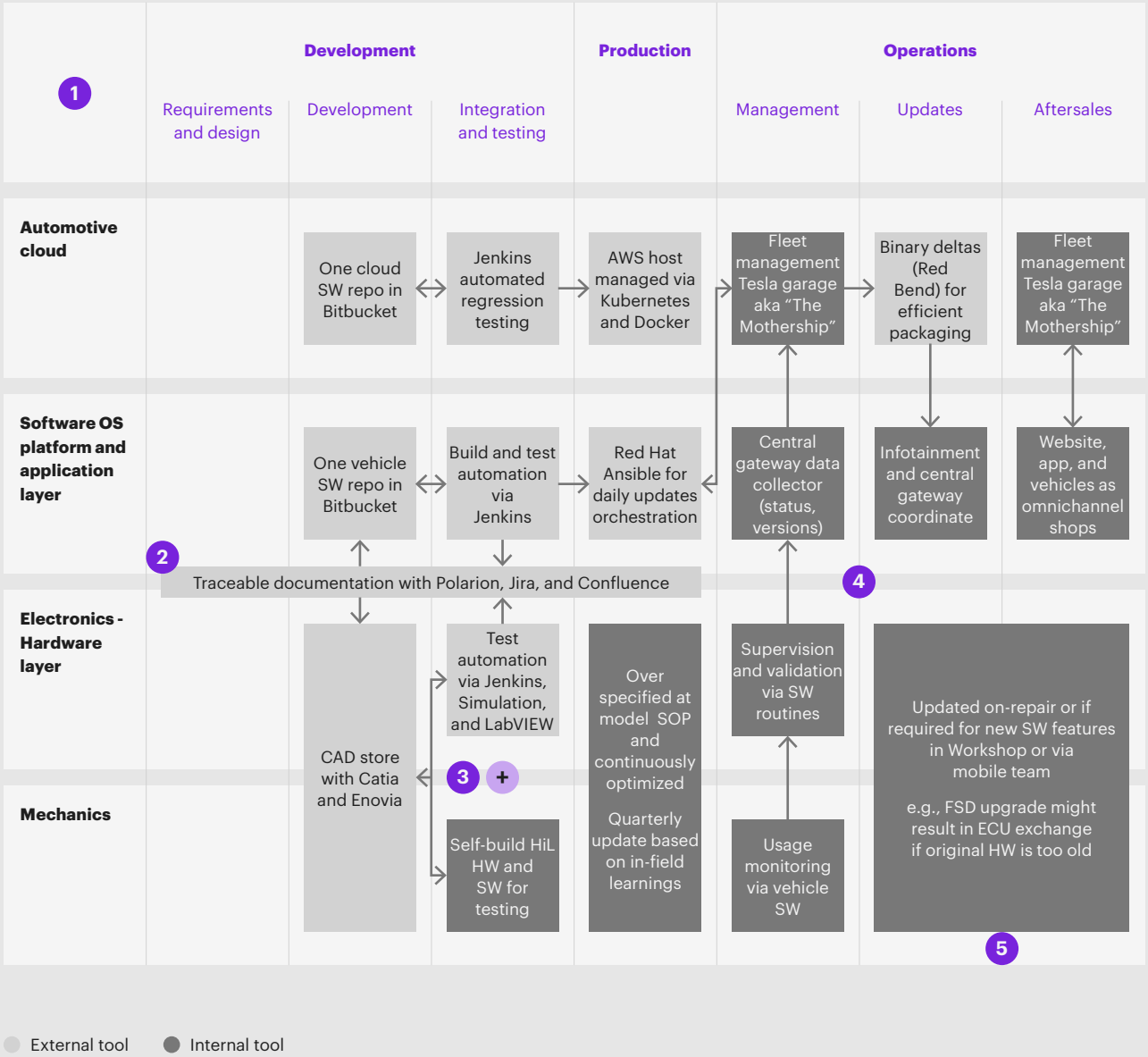
A recurring issue across all five of these approaches is what happens when a new concept is paired with a legacy mindset and organization. In the case of toolchain transformation, the right tool was being used in the wrong way. When organizations implemented a new tool across points of the value chain rather than considering the end-to-end experience, the requirement engineering was not well-connected to development and testing. The common result was a range of tools customized to different workflows that couldn't be merged back together and not the clear and connected portfolio view across one fully automated toolchain.

Figure 5
Approach 5: Tool landscape

Within development a small set of standard tools is leveraged, whereas testing, production, and operations require custom build tools.

What's new

1. **Lean R&D tool landscape** of ~20 development tools
2. **Fully integrated tool chain** for maximum automation and simulation
3. **Central requirements management and tracking** for HW and SW
4. **Internally developed tools** for tasks with no 100% stock fit
5. **50:50 split** in developed and sourced tools



Notes: HW is hardware. SW is software. OS is operating system. HiL is hardwire in the loop. SOP start of production. FSD is full self-driving. ECU is electronic control unit.

Source: Kearney analysis

So, what have we learned?

As we've seen over the past decade, the SDV industry has come a long way. Chief among the many lessons learned is that it's not enough to focus on one aspect of a transformation and that a brownfield approach will fail. Instead, making progress and meeting future challenges demands a comprehensive greenfield approach with an MVP mindset while embracing continuous integration and development (CI/CD) working practices across the whole dev-ops cycle. OEMs know what they want to develop, and going greenfield means they don't have to reinvent the wheel but rather build on proven market standards while reducing customization.

The whole transformation scope must focus on both core functional and soft skills as outlined in this article, which we have summarized below:

— OEMs are more likely to succeed in developing a software culture by carving out lean entities that are independent from the wider organization's models and technology. Successful automotive spin-offs lead to the emergence of the "ambidextrous organization," where one hand sustains the existing business and the other has the freedom to innovate. Any progress made by the smaller software unit can then be brought into the wider business while the degree of separation helps mitigate risk should a project fail.

— Achieving a full-stack view requires a well-planned approach with one domain in focus, also referred to as a "horizontal T" spike. The effectiveness of this approach is clear in cases of neighborhood electric vehicle (NEV) players, such as for autopilot ADAS or customer-centric infotainment—functions that are now seen as brand differentiators.

— Decoupling new technologies from legacy systems is essential. When hardware is tightly connected to software, small changes can have big ripple effects. Decoupling simplifies API management as OEMs are no longer relying on hardware-linked connections. To start a stepwise process of decoupling, organizations should use what works today while capitalizing on the robustness of AUTOSAR and Linux. The transition toward fully decoupled software and hardware vehicle systems will produce a variety of opportunities for OEMs to explore new technologies and shift functionality into future-proofed pure software components.

Contact us if you would like to discuss these insights and learn how we can help assess your level of SDV excellence.

Authors

Sebastian Werner
CEO, BinaryCore
sebastian.werner@binarycore.com

Felix Kreichgauer
Partner, Kearney
felix.kreichgauer@kearney.com

Co-brand authors

Sven Kappel
Vice President; Head of Product Management, Portfolio and Architecture – Software Defined Vehicle, ETAS
sven.kappel@etas.com

Claudio Seitz
Head of Business Strategy and M&A, ETAS
claudio.seitz@etas.com

Markus Menze
Global Head Solution Sales – Software Defined Vehicle, ETAS
markus.menze@etas.com

About ETAS

Founded in 1994, ETAS GmbH is a wholly owned subsidiary of Robert Bosch GmbH, represented in 12 countries in Europe, North and South America, and Asia. The ETAS portfolio includes vehicle basic software, middleware, development tools, cloud-based operations services, cybersecurity solutions, and end-to-end engineering and consulting services for the realization of software-defined vehicles. Our product solutions and services enable vehicle manufacturers and suppliers to develop, operate, and secure differentiating vehicle software with increased efficiency.

etas.com

About BinaryCore

BinaryCore was born from a need to tackle complex technology challenges businesses face in an era when realizing the full value potential of software engineering and cloud usage will clearly distinguish the leaders from the laggards. But addressing these challenges requires radical step change—a willingness to unlearn what went before, an openness to new possibilities and partnerships, and the confidence to make bold decisions and rewrite the rules.

With deep technology expertise, powerful proprietary software, a sharp focus on the numbers, and a clear set of rules, we'll guide you every step of the way to transform your technology and talent set and put in place the infrastructure and teams needed to set you up for long-term success.

binarycore.com

About Kearney

Kearney is a leading global management consulting firm. For nearly 100 years, we have been a trusted advisor to C-suites, government bodies, and nonprofit organizations. Our people make us who we are. Driven to be the difference between a big idea and making it happen, we help our clients break through.

kearney.com

For more information, permission to reprint or translate this work, and all other correspondence, please email insight@kearney.com. A.T. Kearney Korea LLC is a separate and independent legal entity operating under the Kearney name in Korea. A.T. Kearney operates in India as A.T. Kearney Limited (Branch Office), a branch office of A.T. Kearney Limited, a company organized under the laws of England and Wales. © 2023, A.T. Kearney, Inc. All rights reserved.

kearney.com

ETAS | BinaryCore | KEARNEY