THE HIDDEN COSTS OF SOFTWARE-DEFINED VEHICLES

A practical approach to vehicle OS development

White paper

From millions to billions

Global automakers are spending well over $1 billion per year on software research and development, representing $1,000-$3,000 per vehicle sold. But what are they getting for their money?

Due to significant disruption not only to consumer automotive competitive markets, but also to the extensive supporting supply chain, OEMs now dedicate significant resources to software research and development to remain competitive. OEMs are spending their software R&D budgets on a combination of new features and control over their vehicle systems and features.

$1.5-2 Billion – typical annual software R&D spend at global OEMs

With this level of spend, the build or buy decision for software at OEMs needs to be based on a clear understanding of the value each type of software brings to the OEM's ultimate goal of delivering high quality services to their customers.

OEM	2021 Software R&D Estimate[1]	Approximate Per Vehicle Cost	Estimated Software Staffing Commitment
BMW	**$1-$1.5** billion	**$3,000**	3,000-5,000
Ford	**$1.5-1.75** billion	**$1,500**	7,000+ [2]
GM	**$1.5-$1.75** billion	**$1,000**	3,000 new between 2020 and 2021[3]
Mercedes-Benz	**$1.3-$1.7** billion	**$3,000**	4,000-7,000
Toyota	**$3.5-$4** billion	**$1,100**	18,000 across Toyota and all subsidiaries[4]
Volkswagen	**$3-$3.5** billion	**$1,750**	10,000+ by 2025[5]

Table 1 - Examples of OEM software R&D, staffing, and strategic initiatives

These "Vehicle OS" initiatives cover all areas of a software stack – from chip-specific firmware, through the actual device operating system (e.g., Linux, Android, QNX), and up through the abstraction layers, middleware, and application environments, and into the application layers both on and off the vehicle. Every OEM will need to make its own calculations, but for many, different layers of software provide drastically different returns on investment.

Middleware investment in particular presents a conundrum for OEMs – having a great strategy to abstract hardware from applications is critical to deploying a flexible software strategy and working towards a software-defined vehicle. For some OEMs, that strategy may involve authoring a high percentage of their middleware. But as the software costs continue to escalate, many manufacturers may find that partnering or buying is the best approach for their abstraction layers – particularly middleware.

"It is a [dead end] for OEMs to each develop their own operating system."
- Frank Weber, BMW Board Member

High-performance computers as feature opportunity and R&D cost challenge

The emergence of “High-Performance Computers” (HPCs), such as the Continental ICAS used by Volkswagen in its ID range, presents both an opportunity and a challenge for OEMs. These high-powered vehicle components, which allow systems with varying criticality and functionality to operate on the same hardware, are expected to be widely used in digital cockpits, autonomous driving, and core vehicle platform computing systems. The fundamental value proposition of this high-performance computing environment is the ability to dynamically shift computing workloads between the vehicle and the cloud, allowing more advanced computation “at the edge” (in this case, in the car). Developing a solution architecture that allows for this type of device edge computing enables more responsive services while reducing networking and cloud computing costs.

However, software platforms powering edge computation require significant investment to develop. **SBD Automotive estimates that an OEM would need to invest between $65M and $115M over ten years** to develop and maintain the most fundamental, non-differentiating cloud & in-vehicle software building blocks for vehicle data and algorithm management – and this is only a small fraction of the overall software investment required.

Translating these costs across the extensive software-defined vehicle technology stack quickly unveils their hidden development risks of software-defined vehicles: over-investment in enablers that do not directly contribute to consumer differentiation leads to higher costs, longer lead times for development, worse long-term sustainability, and poorer cybersecurity performance. Overzealous ambition to own & develop in-house software platforms injects unnecessary cost in a vehicle’s software bill of materials without creating any value in many cases.

The scalability challenge

The biggest and most ambitious automakers clearly see a future where they own and develop most of the vehicle’s software stack in-house, achieving scalability by developing and sharing this platform through technology-sharing alliances across both internal brands as well as OEM partners.

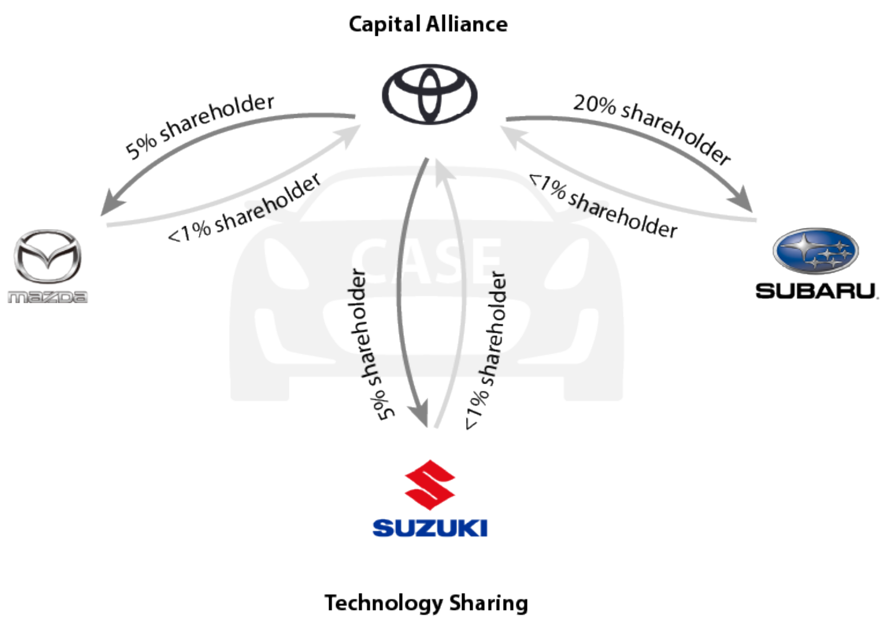


Figure 1 - Toyota technology sharing and capital alliance

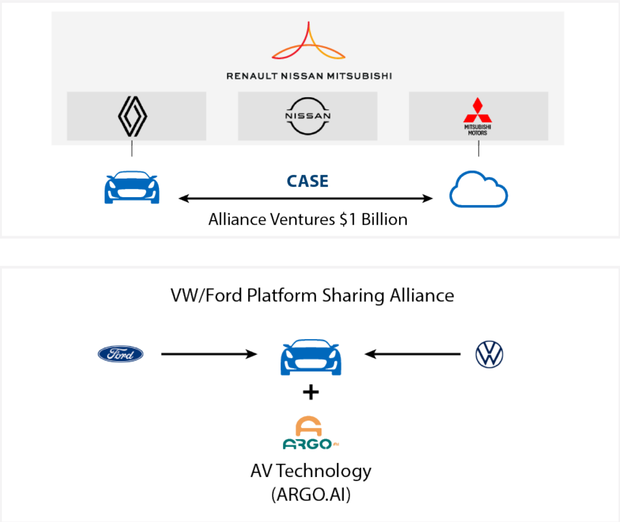


Figure 2 & 3

For example, Volkswagen's CEO, Herbert Deiss, recently noted that their ambition to "own" (i.e., develop, integrate, and deploy) around 60% of the software in their vehicles themselves, representing a sea change from today's vehicles where a significant majority of the software is written by suppliers.

To successfully lead or even compete in the world of software-defined vehicles, OEMs making such investments must have a strategic, objectives-based framework for end-to-end architecture of the vehicle's software stack. Throwing money and people at the problem won't work without direction; using these resources to follow a clear roadmap and build differentiating products and experiences will.

For most OEMs, cost-efficient execution on a software-defined vehicle roadmap will generally require some partner-provided solutions, open-source software, or a combination of the two – particularly for enabling (rather than value-creating) software. Relatively complex software components such as these require only minimal customization for a specific OEM's platform, making it economically inefficient to build the entire stack in-house. In addition, many 3rd party developers seek economies of scale to maximize their addressable market while minimizing investment required to customize solutions for OEMs. Developing custom in-house components from scratch that provide the same functionality as scalable vendor solutions can add anywhere from two to four years to the vehicle platform development roadmap while much of the time resulting in a worse product.

Enabling middleware to decouple software from hardware

Historically, automakers and their suppliers developed software for the unique hardware and device OS environment. This approach has created a complex legacy technology problem, making evolution (rather than revolution) to software-defined vehicles extraordinarily difficult. While this type of software can improve resource efficiency, it also "couples" the software with its specific deployment architecture, making it difficult to update, integrate, and/or scale to additional vehicle platforms or systems.

Middleware is the most important tool for decoupling this software and creating a platform for software that is updatable, scalable, and portable.

Middleware derives its name from its function: to stand in the "middle" between different components or environments. In this case, middleware creates a boundary between the hardware-specific runtime as well as device operating system on a specific component, and the application software deployed to run on the system.

What is middleware?

Middleware provides a standardized 'layer' between software components in a system. It gives developers of systems that integrate with the middleware a set of common interfaces for sending and receiving data or instructions.

Middleware exists to allow for decoupled architectures, meaning software components can be developed and integrated independently, simplifying architectural dependencies in a system and enabling much faster development cycles with simpler testing requirements.

Think of middleware like a job search website like Indeed or Monster. Each software component is like a job seeker or hiring company. Job seekers need not know about other job seekers, and employers don't have direct visibility of other employers. The platform helps connect job seekers and employers. Middleware functionally does the same thing: it simplifies various technical use case by providing a common interface for users.

Figure 4

In practice, middleware bears the burden of understanding the device-specific runtime while translating the deployment-specific nuances to standardized interfaces for higher-level software. This can take the form of APIs (like Toyota's Arene platform), SDKs (like BlackBerry's IVY solution), and/or entire application platforms (like the Android Runtime, AUTOSAR Adaptive Runtime for Applications, and IVY). These full-stack runtimes are engineered to solve specific problems in the vehicle's software architecture. AUTOSAR Adaptive's runtime is usually used to craft domain specific solutions; Android runtime is generally used to host applications (.apks) and provide a rich infotainment UX; and IVY's runtime supports running analytics workloads safely on the car.

Connected middleware: an industry reference

Effectively developing and supporting a software-defined vehicle platform requires a middleware strategy. Because middleware is not a single component but rather a set of software components that provide comprehensive sandboxing and abstraction (that is part of all the runtimes listed above), automakers must take a full-vehicle systems view and derive a software architecture that captures each domain (such as autonomous and digital cockpit) and the associated middleware for each.

As an example of an automotive-specific, connected middleware, BlackBerry and Amazon Web Services (AWS) co-developed IVY to support OEMs to meet these common needs. The application of this platform is outlined below in Figure 5. This figure provides one example of one middleware product (IVY) working together with AUTOSAR and Android middleware components. It offers an example of how OEMs can choose their middleware strategy to include a selection of open-source, standardized, or bought-in components – "owning" the software inventory to the vehicle need not equate to developing everything in-house.

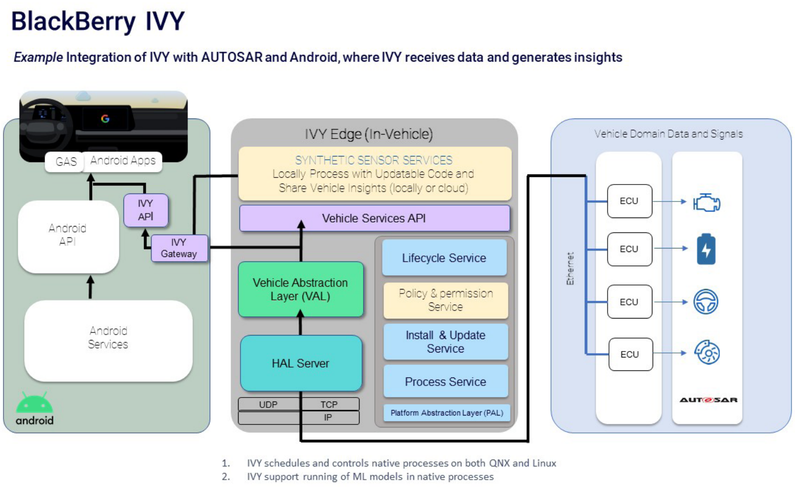


Figure 5 - BlackBerry IVY deployed with AUTOSAR Adaptive and Android Automotive

In this case, IVY's "synthetic sensors" – essentially small analytics applications that work on data or train an algorithm – operate using standardized vehicle sensor interfaces on the vehicle. This architectural pattern provides a homogenous abstraction layer on top of which OEMs, 3rd party developers, suppliers, and other authorized stakeholders can build applications without concern for engineering for hardware or vehicle platform-specific concerns.

By combining common industry middleware solutions such as these, OEMs can achieve a scalable integration showcasing the flexibility of software-defined vehicles.

Balancing cost and risk in vehicle OS development – modelling build vs buy

While the industry faces a spree of software in-sourcing (where OEMs develop vehicle software themselves instead of outsourcing to a supplier), many automakers' immediate reaction to the design requirements for these components may be to design and develop it themselves. After all, the differentiating software that runs on top of these platforms – the primary target of OEM software development – is only as capable as the underlying APIs and services that enable them.

The actual decision is not so simple, however, and requires long-term view of what the OEM will need from its middleware componentry. The biggest return on software investment will be in the software that directly enables a new experience, feature, or service that adds value to the customer or the business. Middleware doesn't directly do any this, in spite of its key role in the infrastructure.

Software and platform vendors such as HARMAN (Ignite), BlackBerry (IVY), Continental (CAEdge), Baidu (Apollo) and industry consortia such as AUTOSAR, GENIVI, and Automotive Grade Linux (AGL), see the opportunity that middleware can provide crucial vehicle OS and middleware infrastructure at a fraction of the cost of new development while also significantly shortening the time-to-market for an OEM's software-defined vehicle roadmap.

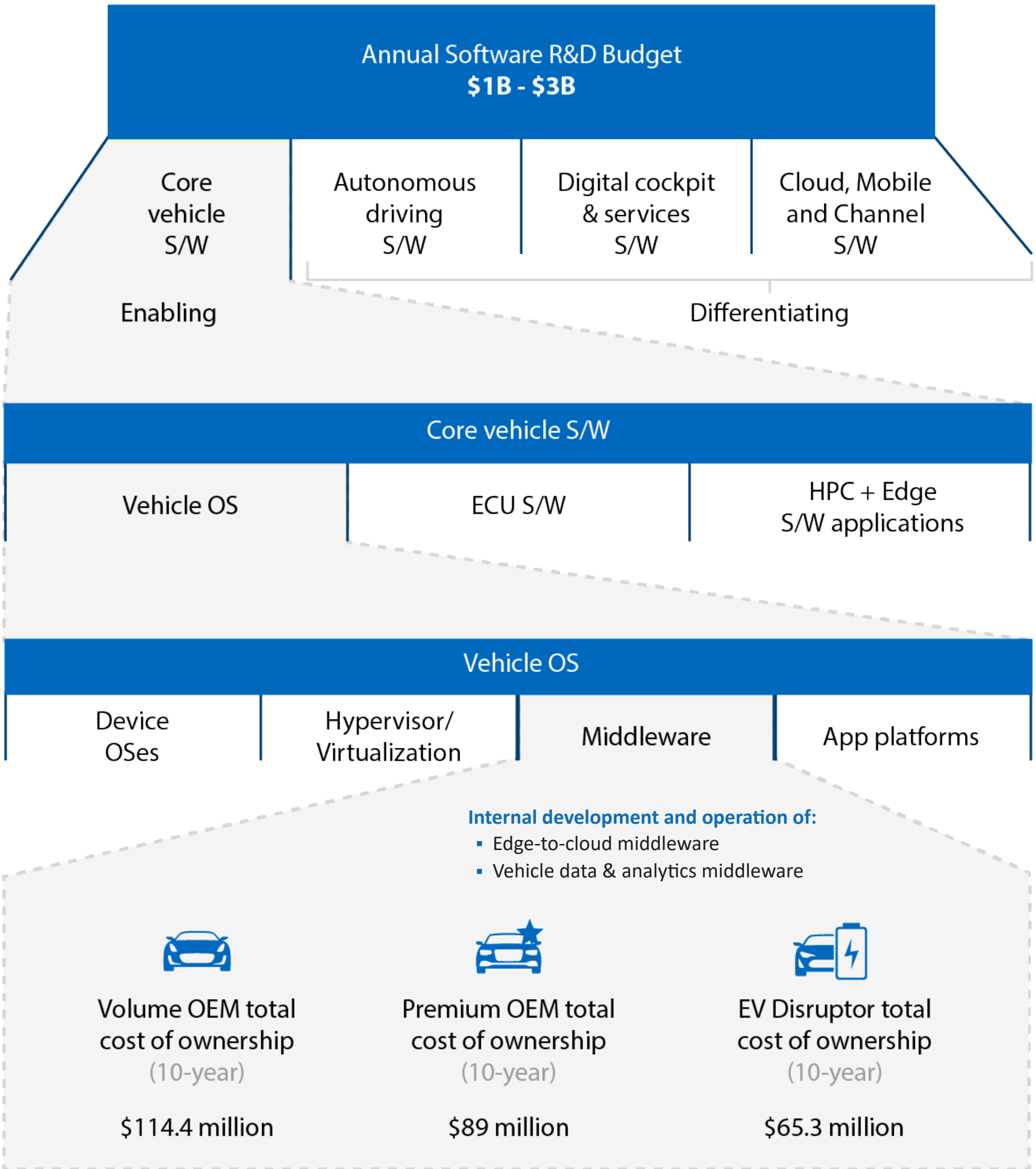
In working with our automaker and supply chain partners, we have developed a shortlist of the most important elements to consider when deciding whether to build or buy core Vehicle OS enablers such as middleware.

The Buy Option: Sourcing vehicle OS infrastructure from software specialists	
The upside:	The downside:
• Significantly lower non-recurring expenses (NRE) and long-term maintenance costs • Stronger security from contractual and/or multi-OEM testing • Access to domain-specific subject matter experts • Scalability of software applications that run on a common multi-OEM platform, resulting in lower cost to add new functions & services from 3rd parties • Instant ecosystem access for tools and partners engineered as part of the provider's middleware stack • Access to pre-existing data services, training algorithms, and (potentially) datasets that shortcut many difficult pain points in autonomous and connected vehicle use cases	• Proprietary software in vehicle stack creates long-term supplier risk • No direct control over product roadmap if new functionality is required • Risk of lock-in on "worse" option if lower cost or open-source alternatives emerge, resulting in problems with legacy software • Potential for unfavorable contracts which preclude the OEM from taking full advantage of the middleware stack without undue expenditure (i.e., usage-based fees)

The Build Option: Designing, developing, integrating, and maintaining an in-house solution	
The upside:	The downside:
• Direct control over how and which data is exposed • Direct control over what security controls are in place and how they are managed • Direct control over the breadth and depth of the supporting toolchain (i.e. SDKs, plug-ins, libraries, Wikis, etc.) • No supplier risk as all software is owned by the OEM • Curated partner ecosystem specific to the OEM's desired brand identity and offering	• Significant up-front non-recurring expenses (NRE) • Long-term time to market (2+ years) • Significant annual expenses associated with further development, maintenance, and re-engineering of the middleware stack • Potentially significant re-engineering expenses for modernization activities • Potential for higher cybersecurity risk from lack of testing and/or robustness • Ground-up ecosystem-building consumes significant time & resources with no guarantee of success

Table 2 & 3

No two automakers are the same, and the strategy for building and integrating a middleware stack should be derived based on the corporate strategy and long-term needs of the business. To help quantify some of the investment required to build a solution from scratch, SBD has developed a model for estimating the total cost of ownership for a middleware stack designed specifically for vehicle-edge processing, data management, and application/edge software lifecycle management.



Distribution of expenses

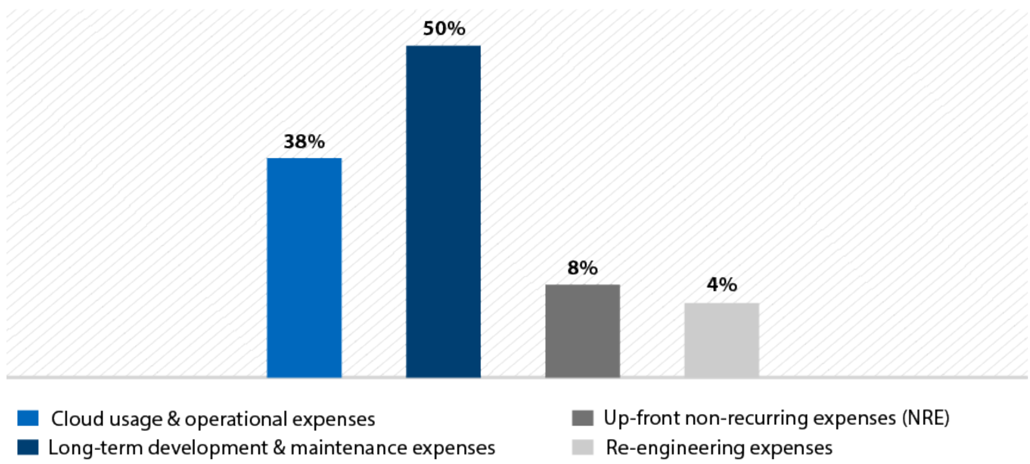


Figure 6 & 7

$115 million is a massive sum for a software project that represents only a small portion of the overall software investment required for software-defined vehicles particularly when considering the middleware software isn't directly providing a monetized feature or service. An OEM may justify such an expense under the pretense that it enables a broader data ecosystem, and the largest volume OEMs are the most likely to undertake such a development effort, although advantages also exist for these OEMs to work with industry partners. But OEMs of all volumes, all margins, and varying brand values should consider a partner-driven approach to mitigate these new development expenses while leapfrogging those competitors who are developing a similar capability in-house over a much longer timespan.

Once launched, OEMs unlock important use cases through their production fleets.

Use Case:	Real-time deployment of a new analysis algorithm for global fleets
Value Proposition:	The ability to support new data-driven business needs or commercial opportunities with little-to-no engineering effort required while reducing the cost to do so by performing data analysis in the vehicle (vs. in the cloud)
Example:	Deploying a new data collection and analysis algorithm to all vehicles enables compressive battery analytics to monitor fleet performance based on a broader set of real-time data (e.g., battery performance, environmental conditions, and driver/vehicle behavior)

Use Case:	Training machine learning models supporting various vehicle components and domains, including digital cockpit, powertrain, autonomous, and body/chassis
Value Proposition:	Continuously improving product quality and performance, resulting in a more competitive product at lower costs while fostering customer loyalty from product enhancements
Example:	Training a machine learning model for optimizing battery range performance based on the characteristics of unique driver profiles

Use Case:	Safe, secure distribution of vehicle data to multiple offboard systems, including OEM-managed as well as 3rd-party managed (such as data marketplaces or service provider partners)
Value Proposition:	Vehicle-specific definition of what data is sent where (enforced on the vehicle runtime itself rather than the cloud), reducing data transmission costs, cloud computing costs, and cloud-based security risks
Example:	Configuring an algorithm which derives four to five key data points relevant a specific insurer and sending them to the driver's selected insurer

Use Case:	Providing an ecosystem on which 3rd parties can develop algorithms or small "applications" that run on any vehicle which implements the target middleware/ edge platform
Automaker Value Proposition:	Access to a pre-developed library of value-add algorithms, partner services, and machine learning models that eliminates the cost to develop the same in-house services or bespoke partner integrations
Developer Value Proposition:	Access to scale and volume with much lower overhead because of non-platform-specific integrations, resulting in a much larger addressable market for their content or service
Example:	A weather company develops a data collection and analysis algorithm which aggregates vehicle-specific sensor data with model data to provide a driver with hyper-localized forecasts and driving conditions

Table 3, 4, 5 & 6

Bridging objectives with middleware design

The question is not if OEMs need middleware to build a software-defined vehicle, but rather how OEMs can integrate and extend it to provide the required enablers most efficiently. The automakers that successfully launch a software-defined vehicle platform with extensible, capable middleware will unlock critical value-adds:

Edge processing of data, resulting in smarter in-vehicle capabilities – especially in the ADAS domain – and significantly optimized over-the-air data costs	**Faster product development cycles** due to enhanced testing and simulation capabilities enabled by abstracted hardware and sensors
Lower-cost, diversified hardware and semiconductor supply chains from hardware-agnostic applications built to run on middleware, not hardware	**Smarter, more personalized in-vehicle experiences** from integrated edge applications providing real-time and predictive services

Table 7

Industry stakeholders must make strategic decisions now on what their business will look like 5 or even 10 years in the future. These choices will have profound impacts on the ability to both execute on software roadmaps and react to competitive disruption. Understanding how to integrate the right middleware for its organization is one of the key decisions that will define what features, functionality, and added revenue streams an OEM can deploy for the next 10 years. Knowing that they need not build everything internally to have control over their software future is the foundation to OEMs making the right choice for their size, capability, and budget.

References

1 SBD Automotive research, aggregating public financial reporting, news reports, and annual volumes
2 https://techstory.in/ford-offers-buyout-for-select-skill-teams-plans-to-attract-software-employees/
3 https://media.gm.com/media/us/en/gm/home.detail.html/content/Pages/news/us/en/2020/nov/1109-tech-jobs.html
4 https://global.toyota/en/newsroom/corporate/35891478.html
5 https://www.volkswagenag.com/en/news/fleet-customer/2021/01/transformers.html

About SBD Automotive

SBD Automotive is a global consultancy firm specializing in automotive technologies. For over 20 years, our independent research, insight, and consultancy has been helping vehicle manufacturers and their partners to create smarter, more secure, better connected, and increasingly autonomous cars. With a reputation for robust data and expert advice, as well as an ability to attract and retain the industry's most talented specialists, SBD Automotive operates a network of local offices in key automotive hubs, including the UK, Japan, North America, Germany, China, and India.

If you would like to learn more, please visit www.sbdautomotive.com