AVNET

DESIGNING TRANSPORTATION SYSTEMS FOR READINESS, RESILIENCE, AND SCALE



JUNE 2026

TABLE OF CONTENTS

Introduction 01

Context 02

Why convergence changes the engineering challenge 03

The four dimensions of converged system readiness 04

Integrated hardware and software 05

Cross-functional system design 06

Development-to-deployment readiness 07

Ecosystem-wide engineering 08

The ramifications for transportation categories 09

Conclusion 10

INTRODUCTION

By 2025, mid-class cars were projected to carry more than 200 sensors for driver-assistance features alone. This statistic is useful not simply as an automotive benchmark, but as evidence of a broader transportation pattern. Across vehicle categories, platforms are taking on more sensing, more software, more communications, and more electronic control. The result is a larger systems burden: more data to move, more heat to manage, more dependencies to validate, and more decisions that have to hold up beyond the prototype stage.

This burden does not stay confined to passenger vehicles. Drones, off-highway equipment, commercial fleets, and other transportation platforms are moving in the same direction, even when their operating environments differ. What they increasingly share is a common engineering problem: architecture decisions now shape far more than performance. They affect data flow, thermal behavior, power strategy, diagnostics, serviceability, sourcing flexibility, and the ability to deploy with confidence.

This is where the second challenge begins. It is one thing to build a platform that demonstrates technical promise. It is another to build one that can survive validation, move cleanly into production, and remain adaptable as requirements, suppliers, and support conditions evolve. This paper focuses on that second challenge. It examines how transportation teams can evaluate readiness, resilience, and scale while key decisions are still fluid enough to improve.

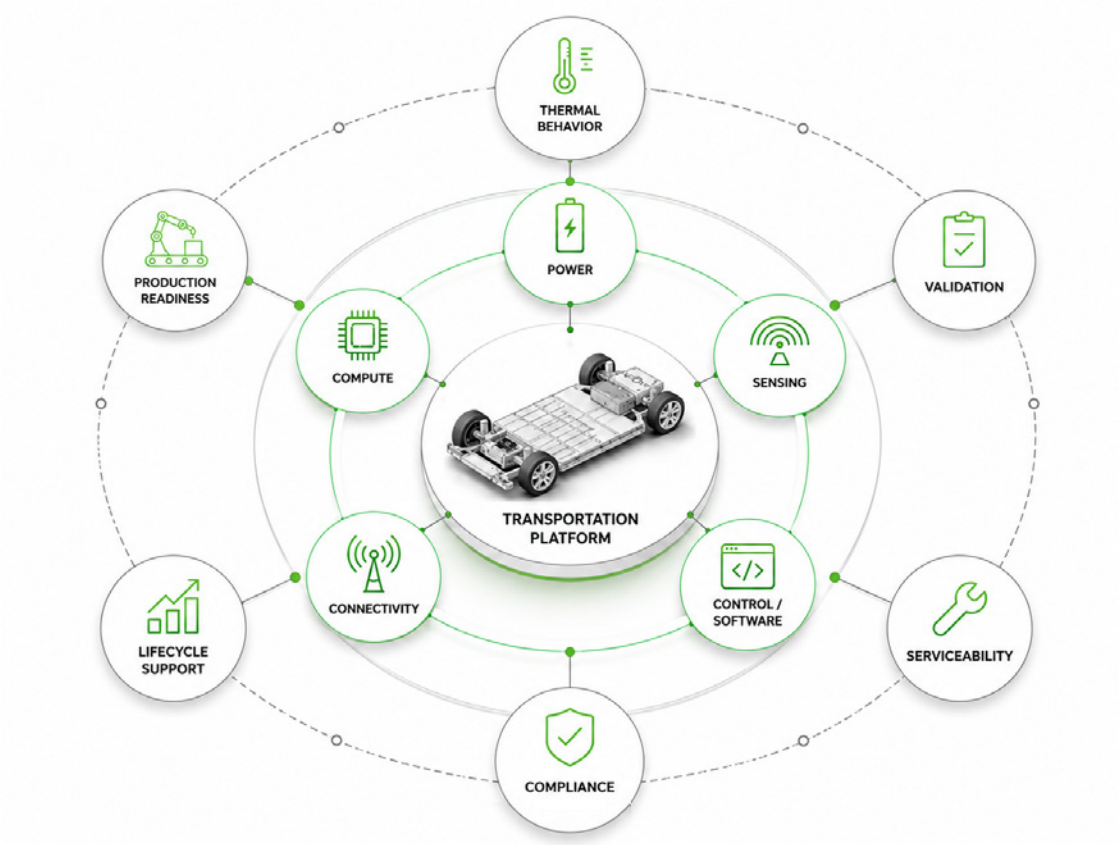


CONTEXT

Transportation platforms are carrying a different kind of systems burden than they did just a few years ago. Electrification ties power architecture more closely to control, thermal behavior, efficiency, and safety. Connectivity extends the design problem beyond launch into diagnostics, updates, and long-term support. Higher expectations for intelligence increase reliance on sensors, software, compute, and real-time data handling. Validation, serviceability, compliance, and regional deployment conditions also have to be considered earlier, because they now shape whether a platform can move cleanly into production at all.

Further complicating the engineering challenge is the tighter coordination those technologies require in real-world operating conditions. A transportation platform may still be described in terms of subsystems, but it behaves more like a connected technical environment. Power affects thermal performance. Compute affects communications load. Sensing influences validation scope. Software decisions carry implications for diagnostics, maintainability, and support. As those relationships become tighter, decisions made in one part of the system become harder to isolate and more expensive to correct later.

This same pattern is showing up across transportation categories that once seemed easier to separate. Passenger vehicles may remain the clearest large-scale example, but similar engineering pressures are now being exposed in drones, heavy machinery, commercial platforms, and lighter mobility systems. The platforms themselves differ. The underlying challenge is becoming more similar: more technical interdependence, more operational exposure, and less room for late correction once programs move toward scale.



WHY CONVERGENCE CHANGES THE ENGINEERING CHALLENGE

Convergence changes the consequence of engineering decisions. A choice that once sat mostly within one subsystem now ripples across the platform, affecting sourcing, validation, manufacturing, support, and scale later in the process. This is the practical shift transportation teams are dealing with. The question is around keeping enough options open to survive real-world program pressures.

A few examples make the point more clearly:

A compute decision can improve performance, while increasing thermal load, power demand, software complexity, and dependence on a narrower supplier set.

A sensing choice can improve functionality, while expanding validation scope, calibration demands, and long-term service complexity.

A connectivity feature can strengthen diagnostics and updateability, while adding certification requirements, maintenance burden, and regional deployment constraints.

A component that solves an immediate design problem can still create downstream risk if it limits adaptability, second-source flexibility, or lifecycle continuity.

This is why converged systems put more pressure on upstream decisions. Engineering choices carry business consequences earlier in the program. A team may still reach a successful prototype, but discover too late that the design is brittle from a sourcing perspective, difficult to validate repeatedly, or harder to support in the field than expected. In that sense, convergence expands the reach of every major decision. It makes cross-functional visibility more important because local optimization is more likely to create downstream friction.

The challenge, then, is not just recognizing that systems are more connected. It is having a practical way to evaluate which dimensions of readiness have to be managed together before those connections become execution problems.

ENGINEERING INSIGHT

Evaluate each major architecture choice for what it changes downstream — thermal behavior, validation scope, sourcing flexibility, service burden, and long-term adaptability.

THE FOUR DIMENSIONS OF CONVERGED SYSTEM READINESS

Transportation systems have to be evaluated by more than performance alone. As discussed before, a platform may function well in development and still prove fragile when it encounters sourcing disruption, validation demands, deployment complexity, or ecosystem dependencies. Readiness in a converged system has to be understood across several connected dimensions, not a single technical lane.

This paper uses four dimensions to organize the challenge:

Integrated hardware and software — whether the platform is being designed as one coordinated technical environment

Cross-functional system design — whether engineering choices preserve sourcing flexibility and long-term adaptability

Development-to-deployment readiness — whether the design can survive validation, production, support, and ramp without major correction

Ecosystem-wide engineering — whether the system can scale across partners, infrastructure, platforms, and regional conditions

These should not be considered separate workstreams. They are overlapping conditions that expose different forms of execution risk. A system may be well integrated at the hardware and software level, for example, while still being too brittle from a sourcing standpoint. It may be sourceable and technically sound, but difficult to validate or support at scale.

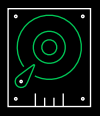
The framework is meant to help teams evaluate earlier whether a platform is likely to hold together when real deployment conditions begin to test it across technical, operational, and ecosystem pressures.

PRACTICAL LENS

Use these four dimensions as an early design screen. If a platform is weak in one, that weakness is likely to surface in the others as the program moves toward scale.

WHERE EXECUTION RISK ENTERS THE SYSTEM

In converged transportation platforms, execution risk usually enters through one of four pressure points:



Integration risk
Hardware, software, sensing, and control do not behave cleanly as one system



Flexibility risk
Design choices narrow sourcing options, regional adaptability, or future updates



Deployment risk
The platform works in development but struggles with validation, production, service, or support



Ecosystem risk
External dependencies such as infrastructure, supplier continuity, or partner technologies are not aligned

Bottom line: Most late-stage program problems are not isolated failures. They are early design choices surfacing under real-world conditions.

INTEGRATED HARDWARE AND SOFTWARE

At the core of a converged transportation platform, hardware and software have to function as one system. While easy to propose, it is decidedly harder to engineer. Compute, sensing, connectivity, and control are often described as separate technical layers, but in practice they shape one another continuously. Hardware decisions define what the software can support in latency, memory, throughput, power draw, and thermal envelope. Software ambitions expose hardware constraints earlier, especially in platforms that depend on sensor fusion, real-time responsiveness, and future updateability.

A sensing stack is only as useful as the software interpreting timing, quality, and context. Connectivity features depend on hardware capability, firmware, diagnostics, update logic, and data handling across the life of the system. Compute capacity isn't only a performance variable. It shapes AI workloads, responsiveness, and how much room a platform has for future functionality before the architecture has to be revisited.

When hardware and software decisions are made independently, the resulting friction tends to show up later and at higher cost. Validation becomes more difficult. Updates become harder to support. Serviceability becomes less predictable. Even platforms that look strong at the prototype stage can become brittle if the hardware-software relationship is not treated as a design requirement from the beginning. This is why integrated hardware and software is not simply a design preference. It is a readiness requirement for any transportation platform expected to perform, evolve, and scale over time.

Even a tightly integrated platform can still run into trouble if the architecture is optimized only for technical performance. In converged transportation systems, platform behavior is defined across hardware and software together, which means weak coordination at that level creates downstream friction that becomes harder to solve later.

DESIGN IMPLICATION | Treat compute, sensing, connectivity, and control as one coordinated design problem early, before validation, updates, and field support expose the gaps.

HARDWARE-SOFTWARE FRICTION POINTS

Examples

- Sensor timing and interpretation do not align cleanly under real operating conditions
- Compute headroom is consumed faster than expected as software scope expands
- Connectivity features work in development, but create heavier diagnostic or update burdens in deployment
- Hardware choices constrain future software features, OTA support, or service access
- Thermal behavior changes once sensing, compute, and communications loads operate together at scale

Bottom line: The problem is rarely one component. It is the interaction between them under real system load.

CROSS-FUNCTIONAL SYSTEM DESIGN

Cross-functional system design is where technical ambition meets real-world flexibility. In a converged transportation platform, architecture decisions shape performance, sourcing options, regional adaptability, certification paths, service complexity, and how much room the program has to absorb change later. This is why design and procurement can't operate as cleanly separated stages. By the time sourcing constraints appear, the architecture may already have limited what the team can do about them.

Some choices tend to have outsized downstream impact:

Power architecture can improve efficiency and system behavior, while also affecting component availability, service complexity, and reuse across platforms.

Wireless and communications standards can simplify connectivity in one market, while complicating certification, regional deployment, or partner integration in another.

Compute platform choices can narrow second-source flexibility if the software stack and interfaces become too tightly bound to one path.

Interface and standards decisions can support modular updates and substitution, or lock the program into brittle dependencies that are expensive to unwind.

This is why resilience has to be treated as a design property. Teams that account for sourcing realities earlier leave room for more options later. Teams that optimize narrowly for footprint, speed to prototype, or peak performance can still end up with platforms that are harder to source, harder to adapt by region, or more vulnerable to supplier disruption. Recent industry survey data underscores the point: **68% of automotive companies report that they are actively restructuring their supply chains using strategies such as near-shoring, friend-shoring, and local-for-local production.** This makes architecture more consequential, because design choices at early stages determine whether a program can take advantage of supply flexibility later and how much resilience or long-term adaptability remains once constraints begin to harden.

WHERE ARCHITECTURE CHOICES CAN NARROW FLEXIBILITY

Examples

- A tightly optimized design leaves little room for substitution if availability shifts
- A communications choice creates new certification or regional deployment limits
- A power architecture improves efficiency but complicates service or reuse
- A proprietary interface improves integration in the short term but reduces adaptability later

Why it matters

Supply flexibility is becoming a strategic requirement. The platforms best positioned to benefit are the ones designed with enough architectural adaptability to absorb change.

DESIGN IMPLICATION

Preserve optionality in the architecture while it is still inexpensive to do so. That includes interfaces, standards choices, supplier flexibility, and regional adaptability.

DEVELOPMENT-TO-DEPLOYMENT READINESS

While it might sound obvious, a transportation platform is not ready because it works in development. It is ready when it can move through validation, manufacturing, service, and sustained operation without major correction. This is where programs begin to separate. A prototype can prove core functionality while still concealing thermal instability, test-repeatability issues, manufacturing difficulty, service burden, or support assumptions that do not hold up under real deployment conditions. This is why prototype success does not guarantee deployment readiness.

The pressure points are usually familiar:



Validation: a system performs in controlled testing, but behavior shifts under broader operating conditions, environmental variation, or edge cases.



Manufacturability: a design meets technical targets, but introduces tight tolerances, integration complexity, or assembly steps that do not scale well.



Serviceability: diagnostics, updates, or component access become harder than expected once the platform is in the field.



Supportability: the platform can launch, but lifecycle maintenance, firmware stability, or supplier continuity introduce hidden fragility.



Production ramp: problems tolerated during development become slower and more expensive to correct once the program is moving toward volume.

A recent example shows how visible this gap can become. A leading automaker delayed production of a next-generation vehicle so it could spend more time on software development and testing, and later reporting indicated that delivery timing had slipped because of issues with new software. The point is not that software problems are unusual. It is that launch readiness in next-generation transportation platforms is intertwined with hardware, software, validation, and deployment timing much more tightly than before.

That broader pattern shows up in recall data as well. NHTSA's 2024 Annual Recall Report says **nearly 80% of vehicle recalls had a final remedy available within 60 days,** which highlights how strongly deployment outcomes are tied to whether issues can be resolved quickly once systems are in market.

Even a platform that is technically ready for deployment can still run into trouble if the surrounding ecosystem is fragmented. The technical implication is straightforward: the farther a readiness problem travels downstream, the more exposed the program becomes to cost, delay, and operational disruption.

ENGINEERING INSIGHT

A design is only as deployment-ready as its weakest transition point, whether validation, manufacturability, serviceability, or long-term support.

ECOSYSTEM-WIDE ENGINEERING

A transportation platform should operate within a broader system of suppliers, software platforms, communications infrastructure, service models, and regional deployment conditions that can either support the program or start constraining it. This makes ecosystem fit both an engineering and operational concern. A platform may be technically strong in isolation but struggle if supporting networks vary by region, if partner technologies introduce integration friction, or if long-term support paths are less stable than the design assumed.

This is one of the more difficult aspects of convergence because the dependencies are not always visible in early stages. Internal engineering teams can control architecture and component decisions, but they do not fully control infrastructure maturity, supplier continuity, certification environments, or how well adjacent systems will stay aligned over time. As platforms become more connected and more software-dependent, those external variables matter more. Update paths, diagnostics, interoperability, and partner compatibility all start shaping whether the product can be deployed and sustained with confidence.

Ecosystem-wide engineering is about designing with external conditions in view. It asks whether the platform can operate cleanly across partners, regions, infrastructure assumptions, and long-term support realities. This broader perspective helps teams surface hidden constraints earlier, before they appear as deployment friction, service complexity, or scale limits. It also helps explain why this challenge cuts across the transportation market, even when vehicle types and operating conditions differ.

DESIGN IMPLICATION

Map your external dependencies early, including infrastructure, supplier continuity, partner technologies, and regional certification, and identify which ones the architecture can accommodate and which ones it can't.



THE RAMIFICATIONS FOR TRANSPORTATION CATEGORIES

The platforms are different. The engineering pressure is becoming more similar. That is the point.

Passenger vehicles may center the discussion because they show the trend at scale. Drones make the same pattern visible in a different form. Heavy machinery brings it into harsher environments. Commercial fleets raise the stakes on uptime and service. Consumer mobility compresses the same technical demands into smaller, more cost-sensitive platforms.

What changes from category to category is the context.



Passenger vehicles push hard on sensing, compute, software, and in-vehicle communications.



Drones put more pressure on weight, power efficiency, and real-time control.



Heavy machinery leans harder on durability, serviceability, and long operating life.



Commercial fleets make uptime, diagnostics, and lifecycle support central.



Consumer and lightweight mobility operate under tighter cost, size, and energy constraints.

The shared issue sits underneath all of them. Teams have to make architecture decisions earlier. They have to preserve sourcing flexibility longer. They have to think harder about deployment conditions before the design hardens. They have to treat support, updates, and ecosystem fit as engineering realities instead of downstream cleanup.

This is why convergence is a critical consideration across the transportation vertical. It does not erase the differences between vehicle types. It reveals a common systems challenge—how to build platforms that can perform under load, adapt under constraint, and scale without becoming brittle, while turning readiness, resilience, and scale into program-level decisions with business consequences.

KEY TAKEAWAY

Different transportation platforms solve for different use cases, but they draw from the same systems logic.

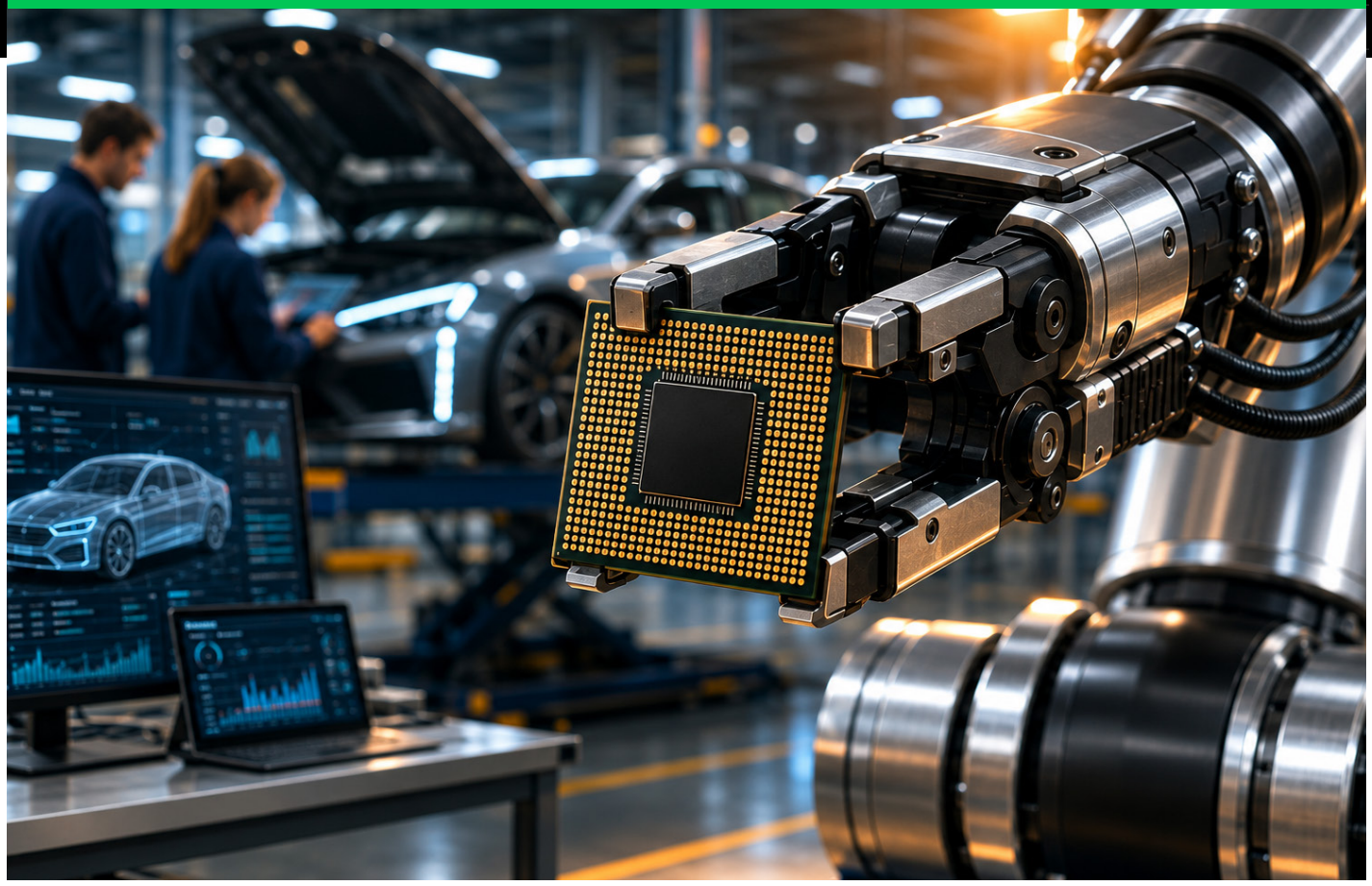
CONCLUSION

Scale is a systems decision.

This is the clearest implication of convergence across the transportation vertical. Programs can break down because the technology is complex, but they also break down because too many critical decisions are made without enough visibility into what the full platform will demand later. Compute choices shape thermal behavior. Sensing choices affect validation burden. Connectivity decisions influence diagnostics, updates, and support. Architecture decisions determine how much sourcing flexibility remains when conditions change. By the time those pressures surface in production or deployment, the room to respond is usually smaller and the cost is much higher.

This is why readiness, resilience, and scale have to be designed in early stages. They are not downstream outcomes. They are the result of engineering decisions made while the platform is still taking shape. Transportation teams that recognize that earlier are in a better position to reduce redesign risk, preserve flexibility, and move toward deployment with more confidence.

This is also where experience matters. Avnet's role in this environment is to help teams think across the platform, understand the trade-offs earlier, and connect design choices to the realities of sourcing, validation, deployment, and long-term support. In a market where transportation systems are becoming more interdependent, that kind of technical perspective becomes part of the engineering advantage.





ABOUT AVNET

As a leading global technology distributor and solutions provider, Avnet has served customers' evolving needs for more than a century. Through regional and specialized businesses around the world, we support customers and suppliers at every stage of the product lifecycle. We help companies adapt to change and accelerate the design and supply stages of product development. With a unique viewpoint from the center of the technology value chain, Avnet is a trusted partner that solves complex design and supply chain issues so customers can realize revenue faster.

Learn more about Avnet at **www.avnet.com**