Qt Group

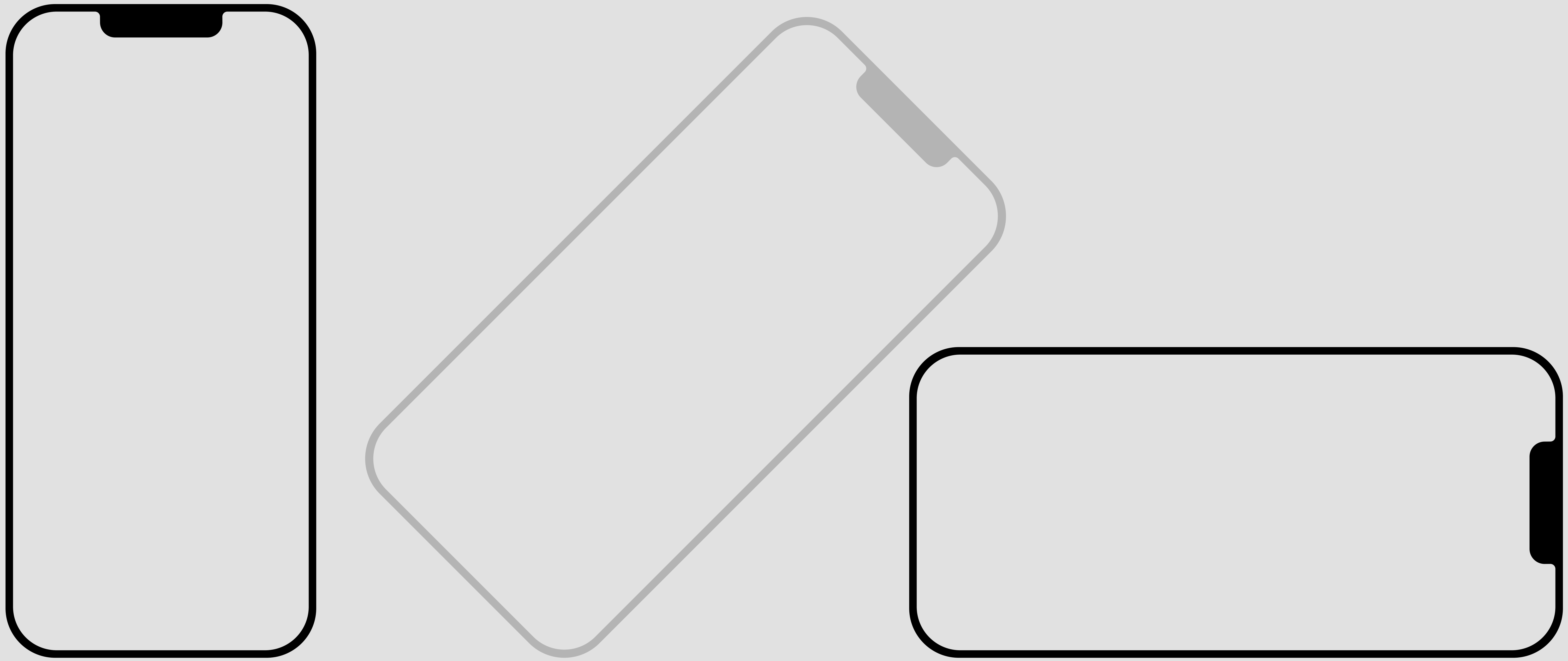
Software-Defined Vehicles Need Software-Defined Leaders

A Strategic Playbook for Established OEMs on Mastering Software Complexity and Leading the Software-First Transformation

2025

Qt Group

Reading on a mobile device?
Turn it sideways for the best reading experience.



Qt Group

Foreword

This playbook will show you how to turn your biggest challenge, software complexity, into your strongest competitive advantage.

As a leader at an established OEM, the core challenge is no longer keeping pace with new entrants, but mastering the immense software complexity that now defines your products.

Transformation is no longer a question of if, but how. This playbook provides the roadmap for building the capabilities you need to win in this reality.

What This Playbook Is

A strategic framework for transforming established automotive organizations into software-first companies. Not by abandoning your strengths, but by applying the same engineering excellence that revolutionized manufacturing to the discipline of software development.

What You'll Master

Confronting the Core Challenge

How the "100M-Line Tech Debt" is the variable impacting your margins, innovation speed, and competitive position today, and the strategic approaches required to manage it.

The Platform Engineering Discipline

How to reorganize software development from fragmented projects into a unified capability. You'll learn why standardization accelerates development and how to make quality happen by default through golden paths and integrated workflows.

Building Your Software Foundation

What technical stack is needed for secure SDV development, from architecture verification to automated compliance.

Your Transformation Roadmap for Tomorrow's Opportunities

How platform ownership enables you to meet evolving market demands (eg. AI safety compliance). Expert guidance on a four-step progression for organizational transformation, with proven strategies already generating returns for OEMs who've made the shift

Who Needs This

Senior Technical Experts and Engineering Leaders who recognize that incremental improvements won't close the software gap. This is for decision-makers ready to make systematic changes to how their organizations perceive, build, and deliver software.

Join the leaders through a strategic transformation that builds on your existing capabilities. The automotive industry has already been disrupted. This playbook tells you how to become the disruptor.

Qt Group

Contents

Qt Group

Key Terminology

Before we dive deeper, let's clarify the most relevant concepts that will guide your SDV transformation:

Software-Defined Vehicle (SDV)

A vehicle where core functionality, features, and performance characteristics are primarily determined and continuously **enhanced through software** rather than fixed hardware. Think of it as a computer that happens to have wheels, rather than a car that happens to have computers.

Platform Ownership

The control over the critical software layers defining your vehicle's functionality, user experience, and future capabilities. Platform Ownership exists on a spectrum and does not narrow down to exclusively building everything yourself. It also refers to having the architectural control to integrate, modify, and evolve your software stack independently of suppliers.

Platform Engineering

A strategic approach to software development that **organizes all toolchains and workflows into "golden paths,"** allowing developers to self-service without building required foundations from scratch. It treats your internal development infrastructure as a product.

Engineering Platform

The concrete solution created through Platform Engineering, **a unified set of tools, workflows, and standards that accelerate development.** Not to be confused with hardware/software platforms where applications run.

Technical Debt

The accumulated complexity and shortcuts in code that make future changes progressively more difficult and expensive. In automotive, this often manifests in the middleware at the source of the multi-millions of lines of legacy C/C++ code, complicating compliance and safety regulation.

Architecture Verification

The practice of continuously checking that software implementation matches its intended design, preventing "architectural erosion" that creates integration problems, and safety and compliance risks.

Qt Group

Introduction

Every OEM Is Now a Software Company

The automotive organizations that master the transformation of becoming a Software company will define the next decade of mobility innovation. Those who defer will find themselves licensing capabilities from competitors who didn't hesitate.

So the question to ask is: who controls your vehicle's future — your engineers or the complexity of your codebase?

Modern vehicles ship with over 150 million lines of code, with projections for a tenfold increase in the coming decade. To put this in perspective, **NASA's** Apollo 11 operated on roughly 145,000 lines. Your vehicles now contain more software complexity than the systems that took humans to space.

This reality has forever transformed the automotive industry. While emerging OEMs, like **Tesla, BYD, Geely,** and others, treat vehicles as software platforms with rapid iteration cycles, established OEMs are in a common struggle with the weight of legacy architectures and complex supply chains.

Platform Ownership = Owning The Future of Your Products

Platform ownership exists on a spectrum. It's not about building everything, but controlling the critical layers that define customer experience and enable innovation. You can benefit from components from different vendors and suppliers, and still "own your platform", if you control the architecture and integration. Then you're building your unique user experience on top. When you own how everything connects and evolves, you own your competitive advantage.

Qt Group

The new competitive landscape is defined by a common metric: **the ability to ship high-quality software at speed**

In 2024, **Recall Masters** revealed in their report that over 28 million vehicles were affected by recalls, with 174 campaigns affecting 13.8 million vehicles linked specifically to software and electronic system failures.[1]

With defects found post-release costing up to 100 times more to fix than those caught during development, software quality is no longer just an engineering concern, but a leadership-level business risk.

The question isn't whether this transformation is happening, but whether you'll master it or be mastered by it. This playbook provides a strategic roadmap for regaining control, turning your hidden liability into a competitive differentiator.

[1] Recall Masters State of Recalls 2024 ↗

1

Legacy Approaches Are Still Failing in the SDV Era

Legacy Approaches Are Still Failing in the SDV Era

You've heard it for years: legacy approaches are failing. For years, industry leaders have recognized that the old way of working falls short for the software-defined era.

But knowing the problem doesn't make it go away. For many established OEMs, the old way of working is still a daily reality and a persistent drag on innovation known as the "100M-Line Tech Debt."

It's a persistent challenge that makes every new project harder and more expensive. It represents not only years of accumulated technical complexity but also the daily frustrations your teams are forced to endure.

This tech debt stems from a fragmented foundation rooted in legacy vehicle architectures. For example, your teams are dealing with a foundation weighed down by dozens of separate, siloed ECUs from different suppliers. This fragmentation makes vehicle-wide software updates—the core of SDV capability—nearly impossible.

This creates compounding problems:

"Legacy architecture is the core problem. A single vehicle has approximately 60–90 siloed ECUs, each acting as a black box from a different Tier-1 supplier, all trying to communicate over slow, low-bandwidth networks.

For an OEM, this fragmented model makes it virtually impossible to build rapid, vehicle-wide FOTA and SOTA updates that modern SDV-makers are known for.

It creates a vicious cycle: When it's time to add new features, the default path has been to just add more siloed ECUs. Each addition makes the entire codebase bigger, more complex, and even harder to update with each model year."

Miao Luo
Director of Technology Strategy,
Qt Group

Driving Up Costs

Software issues now drive a massive portion of vehicle recalls—and fragmented architectures make these issues both more likely and more expensive to fix. The global automotive cybersecurity market size was valued at USD 3.5 billion in 2024 and is projected to grow at a CAGR of 11.6% between 2025 and 2034.[2]

Blocking Future Revenue

Poor architectural control makes deploying reliable Over-the-Air (OTA) updates extraordinarily complex. **Tesla** pioneered OTA in 2012, yet 13 years later, it remains "still new territory for most automakers," according to **Ian Riches**, VP of automotive practice at the analyst firm TechInsights.[3]

[2] GM Insights Automotive Cybersecurity Market ↗
[3] MotorTrend - The Slow Roll of Automotive OTA Updates ↗

"

Software issues now drive a significant portion of vehicle recalls.

"

Dragging On Velocity

Technical debt creates "innovation drag" where simple updates get trapped in undocumented dependencies and integration nightmares. Teams spend more time managing legacy constraints than creating customer value. This creates a massive competitive gap: SDV pioneers ship updates in weeks while legacy teams need 6-12 months for the same changes. This slow pace moves from being an inconvenience to becoming a security risk. When critical vulnerabilities emerge, agile competitors patch their entire fleet via Firmware-Over-The-Air (FOTA) in days, while fragmented architectures leave customers exposed for months.

Without clean and properly managed architecture, OEMs cannot tap into the service-based revenue models that define the SDV era.

Lessons From The Industry

Every new program adds to this technical debt exponentially. Integration challenges multiply as teams try to connect modern systems (e.g. ADAS, and AI functions) with legacy middleware. This reality has led forward-thinking OEMs to reassess their approach.

For instance, **Ford's** strategic decision to refocus their software efforts on mastering their advanced electrical architecture demonstrates the industry's growing recognition that sustainable innovation requires owning the foundational platform first.[4] This change exemplifies how established OEMs are learning to build on their strengths and leveraging decades of engineering expertise while strategically modernizing their software foundation.

[4] CNBC - Ford kills project to develop Tesla-like electronic brain ↗

```
import Quick3DAssets.LaneAssist 1.0
import Quick3DAssets.DangerArrow 1.0
import QtQuick.Timeline
import Data 1.0 as Data

  // Turn signal state
  property bool signaling: laneAssist.lane < 0
  property bool turning: "Turning" === state

  property alias lanelines: lanelines
  property alias timelineAnimationAdas: timelineAnimationAdas

  environment: sceneEnvironment

  // Lighting modes
  Keys.onDigit1Pressed: dynamicLighting.state = ""
  Keys.onDigit2Pressed: dynamicLighting.state = "Day"
  Keys.onDigit3Pressed: dynamicLighting.state = "Dusk"

  // Weather modes
  Keys.onDigit4Pressed: currentWeather = 0
  Keys.onDigit5Pressed: currentWeather = 1
  Keys.onDigit6Pressed: currentWeather = 2
  Keys.onDigit7Pressed: currentWeather = 3
```

2

A Software Discipline for Hardware Companies

A Software Discipline for Hardware Companies

OEMs are struggling to compete using hardware-era processes. When software becomes your primary value driver, a new discipline is required, built on reorganizing your entire development approach.

For decades, established OEMs have perfected the discipline of manufacturing. A software-first company applies that same rigorous mindset to its code, treating quality and architecture as a single, continuous process.

This shift is necessary because software operates under three non-negotiable realities fundamentally different from hardware:

- Software changes constantly
- Software problems multiply, not add
- Software enables continuous improvement – if you control it

You cannot fight these systemic challenges with isolated tools. Escaping their gravitational pull requires a strategic commitment to the discipline built for this new reality: Platform Engineering.

Why Platform Engineering Works Where Others Fail

Think of Platform Engineering as applying the discipline of your vehicle assembly line to a new kind of factory: your software factory.

Just as a physical assembly line relies on standardized tools and repeatable processes, an engineering platform provides developers with standardized tools, templates, APIs, and best practices. The goal is the same: to streamline development, reduce complexity, and enhance productivity. It creates a reliable, consistent environment where teams can build innovative features rapidly, without having to reinvent the foundation every time.

It's a discipline, not a single tool or technology. Just as lean manufacturing transformed how cars are built, Platform Engineering transforms how software is built.

Learn more about Platform Engineering fundamentals ↗

The Proof Is In The Data

Adopting a platform engineering strategy is the most effective response to the challenges of modern automotive software development. To support this approach with hard data, **Qt Group** commissioned a 2024 **Forrester Consulting** market research that confirmed the significant business benefits of this shift. The research, which surveyed over 300 embedded software leaders and professionals, revealed several key advantages:

- **59% Improvement in Development Efficiency**
  The top operational gain cited by a majority of respondents.
- **Enhanced compliance and security**
  Meeting strict regulatory requirements while maintaining robust security measures.
- **Reduced Maintenance Costs**
  Lowering the long-term cost of ownership for the software platform.

The study also revealed a key advantage for the automotive industry in particular improved coordination among embedded software development teams, including developers, design-ers, and testers.

Taken together, these data points show that a platform strategy provides a clear competitive advantage, allowing businesses to innovate faster and more safely.

What Platform Engineering Delivers

Unlike traditional approaches that treat quality, development, and deployment as separate concerns, Platform Engineering creates a cohesive approach through a unified engineering platform where:

Standardization doesn't mean rigidity:

Supporting Teams to follow "golden paths" for 80% of use cases while maintaining flexibility for innovation

Quality happens continuously:

Opting for an end-to-end software development solution that includes tools to assure compliance continuously within the development workflow

Architecture stays clean:

Providing tools to continuously verify that implementation matches design, preventing the software erosion that creates technical debt

Reusability becomes default:

Components built once serve many programs, with 35%* reporting faster time-to-market

The shift from fragmented tools to an integrated platform directly attacks the root causes of the "100M-Line Tech Debt" challenge by making it impossible to bypass quality and architectural standards.

For deeper insights into Platform Engineering principles, see:
Laying Golden Paths: How Executives and Engineers Can Align on Platform Engineering. ↗

3

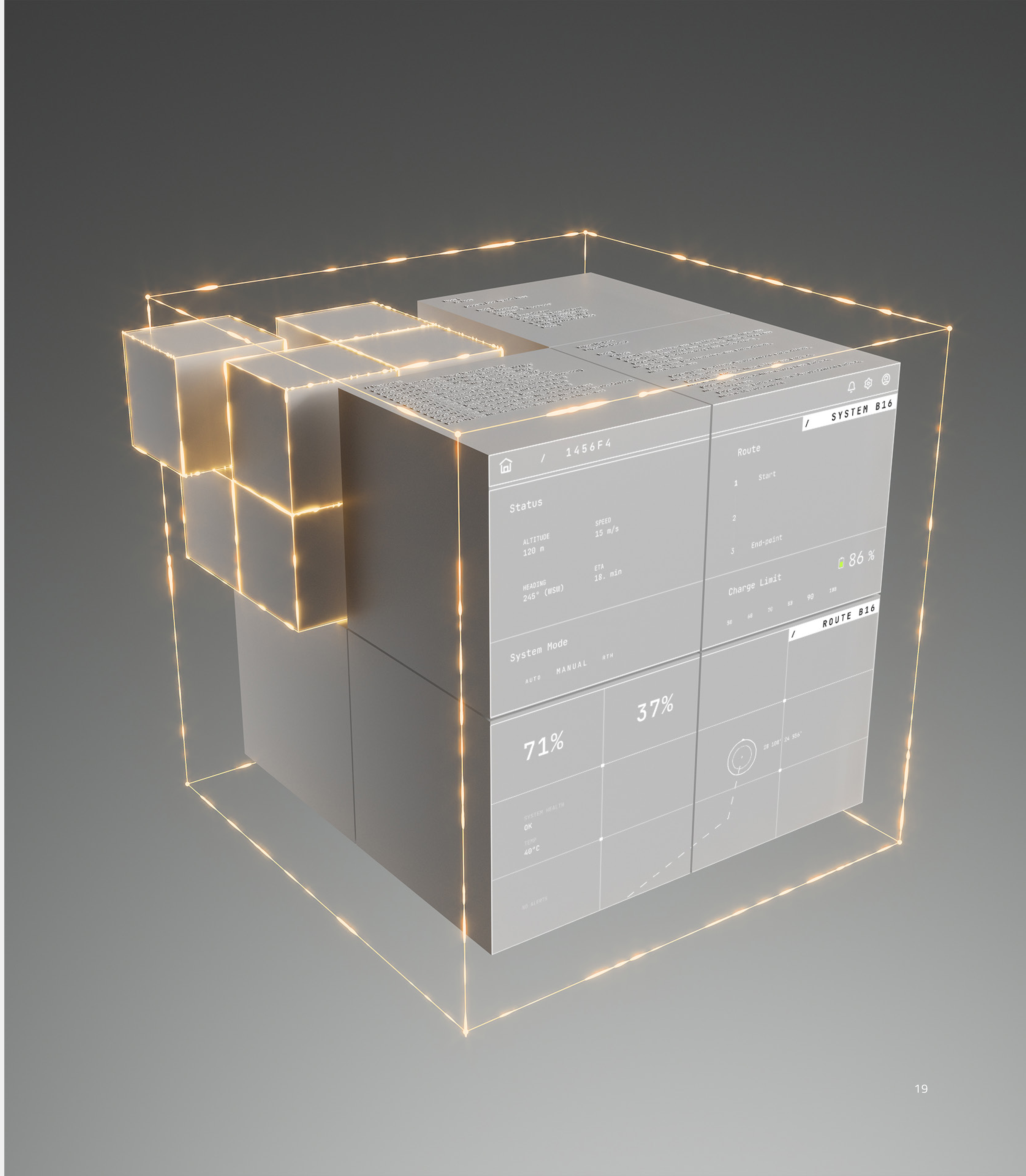
Quality, Safety, and Security Build the Foundation

Quality, Safety, and Security Build the Foundation

Platform Engineering exemplifies a software-first discipline by embedding quality, safety and security throughout the entire development lifecycle. This approach is essential for SDVs and takes you closer towards taking more ownership of your platform.

Quality Starts with Architecture

An effective platform requires more than just good code; it demands an end-to-end solution that unifies software development and quality assurance into a continuous process. This begins with architectural integrity and is realized through an integrated set of modern tools.



The Exact Stack You Need for Building a Secure Development Environment

Modern SDV development requires a secure platform built on two core pillars: a robust development toolchain and a seamlessly integrated quality assurance solution.

Development Toolchain

- **Unit testing framework** to run fast unit tests in CI
- **Profiler** to optimize UI responsiveness and fluidity
- **Static builds** for high-integrity, safety-critical systems, demanding higher control, security, and certifiability
- Detailed **SBOM** to check SW dependencies and promptly act in case of vulnerabilities
- **Long-term Support** with API stability guaranteed for years

Integrated Quality Assurance

- **Architecture verification** to prevent design drift
- **Embedded compliance** for MISRA, ISO 26262, AUTOSAR and others
- **Comprehensive code coverage analysis** with MC/DC support
- **Integrated testing** across GUI, static analysis, and performance

The Compounding Returns of Quality

When quality is part of the development process, rather than bolted on at the end, the benefits compound:

- Defects are caught when they cost a fraction of the potential late finding cost
- Compliance documentation is generated automatically, not through manual audits
- Developers get immediate feedback, not delayed review cycles
- Architecture stays clean and maintainable

During platform engineering transitions, **43%* of organizations cite security and compliance as one of their biggest challenges,** This is especially important for safety-critical automotive applications where development teams hold full lifecycle responsibility from the first line of code until a device is decommissioned for mission-critical and safety-critical applications.

For more on automotive software testing best practices, see
Automotive Software Testing: Ensuring Quality in the Age of SDVs. ↗

From Foundation to Innovation — Real-World Success

The shift to becoming a software-first company demands more than new tools. It requires a fundamental change in mindset. Where hardware once defined the vehicle and software merely supported it, software now defines capability while hardware becomes the enabler.

This transformation requires a discipline where software is designed, built, and tested as the primary value driver. Platform ownership makes this shift concrete and actionable, and it's no longer theoretical — industry leaders are already proving what's possible.

At Qt World Summit 2025, **Harman International** demonstrated how owning their software stack enabled them to deliver differentiated experiences while maintaining the quality and safety their brand demands.

Watch the full Harman case study presentation ↗

A great example that illustrates how automotive manufacturers who are working on having control on their platform who control their platform, from UI framework to rendering engine and beyond, gain crucial advantages such like:

- **Customization without compromise:** Tailor frameworks to specific hardware while maintaining safety standards
- **Future-proof architecture:** Control long-term evolution without vendor lock-in
- **Seamless integration:** Unite proprietary innovations with partner solutions
- **Data-driven enhancement:** Continuously improve based on real vehicle data

KEY INSIGHT

Platform ownership enables the rapid innovation cycles that define SDV success.
When you control the foundation, you control the pace of innovation.

4

Platform and Codebase Control Drives Innovation

Platform and Codebase Control Drives Innovation

Platform and codebase control solves today's problems while enabling tomorrow's opportunities. Let's examine three important areas where architectural control enables breakthrough capabilities.

Mastering AI and ADAS with Architectural Control

While the industry focuses on autonomous driving, AI is transforming every aspect of vehicle software, and multiplying its complexity. For development teams, this creates a dangerous blind spot for safety and compliance, and a critical quality assurance challenge for an industry built on strict safety regulations.

For example, AI and ADAS features rely heavily on GPU-accelerated code that traditional static analysis tools cannot analyze, making it impossible to prove functional safety for code your existing quality tools can't even see.

Consider this: **Gartner** predicts that by 2027, 17% of cyberattacks will involve generative AI.

Without platform-level control and full visibility into the code, how can you protect against these emerging threats?

Architectural control with deep code analysis solves these challenges. It allows you to prove ISO 26262 compliance for AI-driven systems, deploy ADAS updates with architectural integrity, and iterate on machine learning models without compromising stability.

[5] Gartner Forecasts Global Information Security Spending to Grow 15% in 2025 ↗

Struggling with ISO 26262 compliance for your complex systems?
Our guide provides a clear path forward.

KEY TAKEAWAY

AI represents the next frontier of automotive innovation but requires platform-level architecture control to implement safely and effectively while staying compliant.

Mastering the Android Automotive Experience

As Android Automotive OS becomes a popular option for OEMs, the new challenge is ensuring implementation quality and brand differentiation on a standardized platform.

This challenge is often rooted in the "black box" model of the supply chain. To protect their valuable intellectual property, suppliers deliver applications as compiled binaries without source code. This is a standard and understandable industry practice. The challenge, however, remains for the OEM, who is ultimately responsible for guaranteeing a cohesive, high-quality user experience across dozens of these integrated systems.

Platform ownership provides the tools to solve this integration challenge by enabling:

- Comprehensive testing frameworks that validate behavior without source code access.[6]
- A consistent user experience across all supplier-provided components.
- True brand differentiation on a standardized platform.
- Architectural flexibility to swap components without rebuilding the entire system.

KEY TAKEAWAY

Platform ownership provides the tools to enforce quality and brand consistency, turning the commodity Android Automotive OS into a unique competitive advantage.

Building a Data-Driven User Experience

The ultimate competitive advantage in the SDV era is a superior user experience that continuously evolves based on real-world data. This data loop, enabled by Over-the-Air (OTA) updates and platform control, is the engine for future revenue and brand loyalty.

The OTA market is projected to grow to over $13 billion by 2032, but only OEMs with the architectural control to manage this data can capture the opportunity.[7]

With a proper platform foundation, your vehicles become learning systems. This allows you to:

- Deploy **performance improvements** that make the vehicle feel more responsive.
- Inform **future feature development** based on how customers actually use the HMI.
- Enable new services like **predictive maintenance** that improve the overall ownership experience.

KEY TAKEAWAY

Architectural control is the key that unlocks a continuous improvement cycle, allowing you to use real-world data to constantly enhance the user experience and capture new OTA revenue streams.

[6] QT - How to Simplify and Scale UI/GUI Testing for Android Automotive ↗
[7] GM Insights - Automotive Over-The-Air (OTA) Update Market Size ↗

CUSTOMER SUCCESS Elektrobit Axivion

How Elektrobit Accelerated Development Through Architecture Control

The Need for Speed Without Sacrificing Quality

Elektrobit's EB street director navigation software powers vehicles from **Audi, Porsche, Mercedes-Benz,** and **Volkswagen.** These premium manufacturers demand exceptional performance and rapid feature delivery.

With over 100 developers across multiple global locations working on a single modular platform, Elektrobit faced the classic challenge: maintaining architectural integrity while shipping new features at speed. New developers needed weeks to understand the system. Every feature risked breaking existing functionality.

Resolved Through Continuous Architecture Verification

Elektrobit implemented a unique solution for automated architecture verification that continuously checks whether code matches its intended UML-based design. Violations surface immediately, not months later during integration testing.

This transparency transformed their development. Teams make informed decisions about speed versus maintenance trade-offs. Developers see how their code affects the overall system. New team members onboard faster through tooling rather than documentation.

CUSTOMER SUCCESS

The Results: Structure Enables Speed

The transformation delivered measurable impact:

- Features ship faster—teams know immediately when changes violate boundaries
- New developers become productive weeks earlier
- Project estimates become accurate based on real architectural impact
- The modular architecture remains maintainable despite continuous evolution

Elektrobit proved that architectural discipline accelerates development. Clear boundaries improved testability, and continuous verification prevented debt accumulation while maintaining automotive safety standards.

KEY TAKEAWAY

Elektrobit demonstrates this playbook's core principle: controlling your architecture means controlling your innovation pace. For OEMs facing distributed teams and legacy systems, continuous architecture verification turns complexity into manageable structure, making quality automatic and innovation repeatable.

Read the full success story ↗

5

The Path Forward – 4 Expert Steps to Transformation

The Path Forward – 4 Expert Steps to Transformation

Success doesn't require abandoning existing investments. Platform transformation can be implemented progressively, delivering value at each stage.

While the exact phases should always be tailored to your specific business goals, successful transformations follow a proven high-level pattern. To outline this roadmap, we spoke with our seasoned expert, **Miao Luo, Director of Technology Strategy,** to share his insights on the four critical steps.

STEP 4

Scale and Communicate Your Success

STEP 3

Develop Skills and a Culture of Ownership

STEP 2

Build a Prototype Organization

STEP 1

Secure Long-Term Leadership Commitment

STEP 1

Secure Long-Term Leadership Commitment

Transformation requires determined leadership commitment: 5+ year roadmap, protected budget, and dedicated SDV champion driving top-down change without disrupting operations.

"A lack of long-term commitment is the enemy of transformation. Many established OEMs have failed at the first sign of trouble when costs skyrocketed. In my mind, true transformation means to suffer with a conviction — a deep certainty that the difficult journey is necessary and will ultimately lead to a much better future for the company."

STEP 2

Build a Prototype Organization

With leadership onboard, the next step is to test SDV concepts without altering core production.

"Form a small, cross-functional 'skunkworks' team to retrofit an existing vehicle model with new SDV features. Start with Software-Over-the-Air (SOTA) capabilities and then move to Firmware-Over-the-Air (FOTA). The goal is a low-risk commercial pilot that proves the concepts in the real world."

STEP 3

Develop Skills and a Culture of Ownership

A successful transformation requires empowering your existing talent and fostering a new mindset.

"Train your people in agile software development, standardize your tools, and foster a SDV mindset. Iterate based on pilot program feedback, which helps build expertise while leveraging existing talent. Most importantly, give your development teams the autonomy to make decisions and execute. A culture of ownership starts with the individual and cannot be just a slogan in an internal press release."

STEP 4

Scale and Communicate Your Success – Internally and Externally

Once the model is proven, the final step is to transition the entire business model while maintaining stakeholder confidence.

"Expand the proven SDV features across your vehicle models and integrate the new processes into your main production lines. It is crucial to market these successes both internally to your teams and externally to investors and customers to build and maintain momentum."

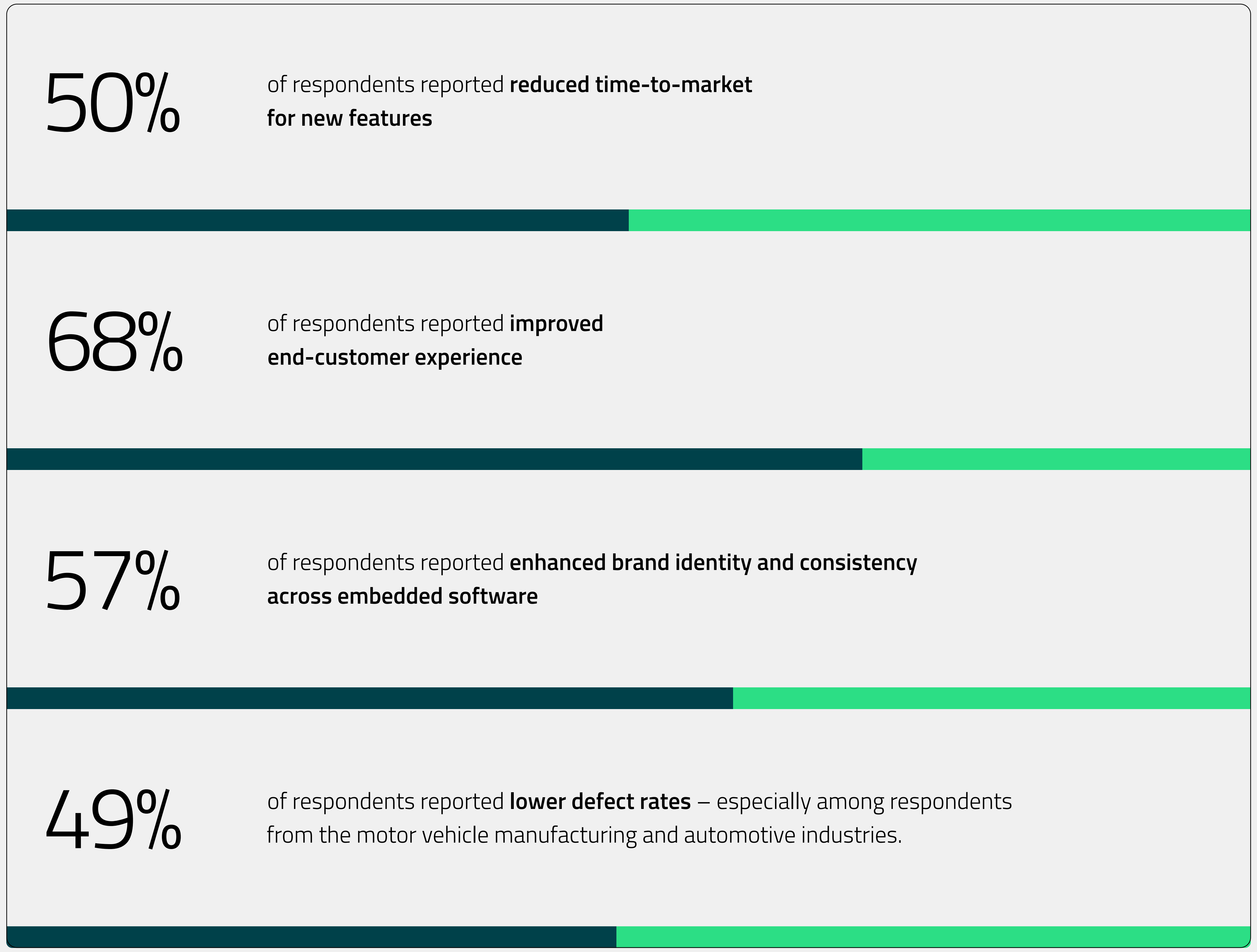
— Miao Luo,
Director of Technology Strategy, Qt Group

The ROI of a Software-First Shift

As stated earlier, the business case for adopting and implementing a software-first transformation is backed with clear data. **Citing back to the 2024 Forrester Consulting study commissioned by Qt Group,** industry leaders were asked about the impact of adopting modern, platform-driven practices.

The top benefits their businesses have experienced or expect to experience include:

For a deeper look into this concept, see
What Big Firms Can Learn from Smaller Teams About Platform Engineering. ↗



50% of respondents reported **reduced time-to-market for new features**

68% of respondents reported **improved end-customer experience**

57% of respondents reported **enhanced brand identity and consistency across embedded software**

49% of respondents reported **lower defect rates** – especially among respondents from the motor vehicle manufacturing and automotive industries.

These impressive results stem from a simple principle: a platform-driven strategy allows large organizations to achieve the agility of a small team, but at an enterprise scale.

Qt Group

Conclusion

Conclusion— The Choice Is Yours

The rules of the automotive industry have been rewritten by software. Competitive advantage now flows from the discipline to deliver superior software, built upon the twin pillars of architectural control and continuous quality.

Throughout this playbook, we've confronted the "100M-Line Tech Debt" as the primary obstacle to innovation. We presented platform ownership and codebase control as the strategic solution and provided an expert's roadmap to guide your transformation.

The path forward presents a clear and urgent choice.

Option 1: Choose to Remain reactive

Continue managing complexity with fragmented, effort-heavy solutions. Watch as software recalls erode margins, security threats damage your brand's reputation, and more agile competitors capture the market with features you struggle to build.

Option 2: Own the Stack and Define the Future

Choose to define the future of mobility. Transform the vehicle from a finished product into a living platform that delivers new value to customers throughout its entire lifecycle. Shift your business model from one-time transactions to continuous, value-driven relationships. Become the technology leader that defines the pace of innovation, not just reacting to it.

OEMs that master this transformation will define the next decade of mobility. Those who defer will find themselves licensing capabilities from competitors who didn't hesitate.

Your Next Move

The automotive industry's transformation is complete. Software defines vehicles. Data drives revenue. Platform ownership determines who leads and who follows.

Schedule a strategic consultation to assess your current platform maturity and build your transformation roadmap.

Schedule a consultation

qt.io/contact-us

Qt Group

About Qt Group

Qt Group (Nasdaq Helsinki: QTCOM) is a global software company, trusted by industry leaders and over **1.5 million** developers worldwide to create applications and smart devices that users love.

We help our customers to increase productivity through the entire product development lifecycle: from UI design and software development to quality management and deployment. Our customers are in more than **70 different industries in over 180 countries.** Qt Group employs some **900 people,** and its net sales in 2024 were **209.1 MEUR.**

To learn more, visit **www.qt.io**