Our heart
beats
embedded.

Five major challenges in software development for automotive microcontrollers

A guide for vehicle manufacturers and suppliers

Empowering Tomorrow's Automotive Software
www.etas.com

ETAS

Abstract

As exclusive enablers of real-time capability and functional safety within vehicles, microcontrollers continue to play a crucial role in the automotive industry. However, optimizing the development process of the microcontroller software is not commonly a top priority. The market is paying more attention to differentiating and innovative applications than to embedded functions working in the background. On the road towards the software-defined vehicle, basic development tasks tend to fall by the wayside. However, it is precisely the optimization of this process that guarantees future success for both vehicle manufacturers and suppliers.

To facilitate a state-of-the-art automotive software development process, we as ETAS have compiled the top five major challenges our customers are facing: high integration efforts, complex calibration, time-consuming testing, limitations in scalability and flexibility, as well as holistic cybersecurity requirements. Mapped to the specific process steps along the V-model, this white paper describes the challenges and provides adequate solutions, which will help you boost the efficiency of your development processes, reduce workloads, implement higher security standards, and ultimately keep pace with the fast-evolving market.

Table of contents

1. Introduction

Safety, real-time capability, resource optimization – these properties largely depend on electronic control units (ECUs), specifically microcontrollers, and are the basis for a variety of important functions in modern vehicles. However, the innovation pressure on original equipment manufacturers (OEMs) and suppliers often causes them to disregard the development processes of ECU software in favor of more exciting technologies. Nevertheless, ECUs offer a lot of potential for improvement: more automation, higher efficiency, overcoming security limitations, to name just a few. As requirements increase and innovation speed becomes a decisive factor, not updating these basic processes will eventually take its toll.

There is a lot of this fundamental work to do during the development of a typical modern vehicle. The largest part of the vehicle's code is spread out on microcontrollers and fulfils its routine job without being noticed by the user. However, this deeply embedded software has a very special responsibility in a vehicle: a malfunction in a car component can have a serious impact on the health of passengers. The requirements from the automotive industry are accordingly high, especially when it comes to fulfilling permanently evolving safety regulations. Although a routine task, software development for embedded microcontroller systems is extremely important – and will remain so for years to come.

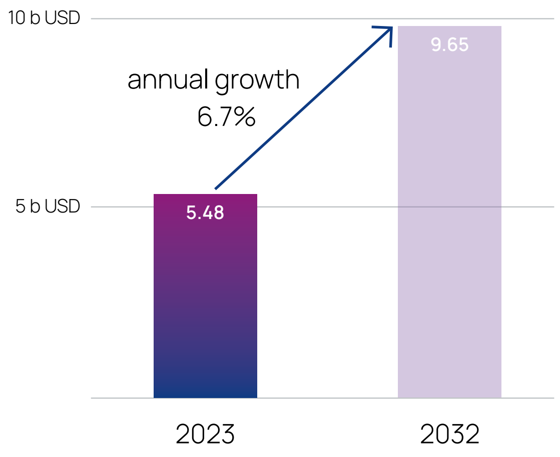
This white paper is designed to help manufacturers and suppliers reorganize their processes towards state-of-the-art microcontroller software development. It sets off with basic information about the importance of vehicle microcontrollers and the steps needed to develop new software for an ECU generation change. It then provides a deep dive into the main topic: the five major challenges during the development process and possible solutions. Not every automotive company is affected equally by all challenges. Some manufacturers and suppliers do not perform all steps in-house. Further factors are just as decisive, e.g. the maturity of digitalization, the composition of existing tools, the amount of legacy software, and the hardware setup. Nevertheless, knowing the big picture, the common struggles, and potential remedies is always beneficial for each player within the automotive industry.

2. A glance into the world of vehicle microcontrollers

Most control and communication tasks in a vehicle are carried out by a variety of microcontrollers as the heart of ECUs: from cost-effective 8-bit variants for sensor interface and control through to 32-bit for modern infotainment displays and vehicle dynamic systems. Standard modern vehicles contain over 40 of these electronical components, high-end variants even up to 150. They ensure the reliable functionality of e.g. airbags, ABS, ESC, controlling engines, tire pressure, and battery status. All this happens in the background, unnoticed by the driver.

Currently, OEMs and suppliers are facing increasingly shorter E/E architecture innovation cycles and a growing number of vehicle variants and functionalities. Hence, microcontroller-based software development must also become more and more efficient. Trends associated with the software-defined vehicle, such as hardware-software decoupling, centralized vehicle computers, or vehicle cloud computing, only seemingly replace the classic ECU approach. In reality, embedded systems will continue to play a decisive role, since they are the only way to ensure real-time capability and functional safety up to ASIL-D. Market reports predict considerable growth in the microcontroller sector (see figure 1). OEMs and suppliers must therefore adapt their development processes for ECU software to the new market requirements regarding speed, flexibility, and security.

Figure 1: Global microcontroller market growth



The global market size for automotive microcontrollers is expected to grow at an annual rate of 6.7%.

Source: https://www.fortunebusinessinsights.com/de/markt-f-r-mikrocontroller-f-r-die-automobilindustrie-104084

Software development for embedded microcontroller systems is neither new nor exclusive to the automotive sector. These components are installed in electronic devices and everyday objects, from toothbrushes to vacuum cleaners. What makes the development process special when it comes to automotives? The most important aspect is the high safety and reliability requirements from the very first minute the vehicle hits the road, followed by the maintainability over decades. Moreover, reliable real-time capability is crucial for many systems like powertrain controls or vehicle dynamics: the electronic control system must always keep pace with the physical processes unfolding within the vehicle or the environment.

Although microcontroller programming is a cyclical routine task for OEMs and suppliers with each generation change, adapting the process to the high regulatory demands while making it as efficient as possible within a global OEM and supplier network is challenging. This is particularly true for companies that use long-established process chains with tools and methods that are only partially compatible with current requirements.

i

Briefly explained:

The harmony of ECU, MCU, MPU, and VCU

Electronic Control Units (ECUs) employed in automotive systems are specialized computers that manage specific functions in the vehicle. Within these ECUs, microcontrollers (MCUs) handle real-time tasks like engine control and sensor management. Microprocessors (MPUs), which are found in more advanced ECUs, provide higher processing power for complex applications such as infotainment and ADAS, often running on operating systems like Linux. The vehicle computer (VCU) coordinates these ECUs, ensuring seamless communication and operation across the entire vehicle network.

3. The vehicle ECU software development process

Although E/E architectures are being reorganized towards a more centralized approach, microcontroller-based ECUs are here to stay. The general development process for the embedded software could theoretically continue as it is. However, the development cycles will accelerate drastically to fulfil the needs of a functionality-thirsty market, as we are encountering with the trending software-defined vehicle (SDV). Traditional approaches cannot keep up with that speed. They can, however, also not simply be replaced overnight. In many cases, the optimal solution consists in combining tried and tested processes and tools with innovative solutions.

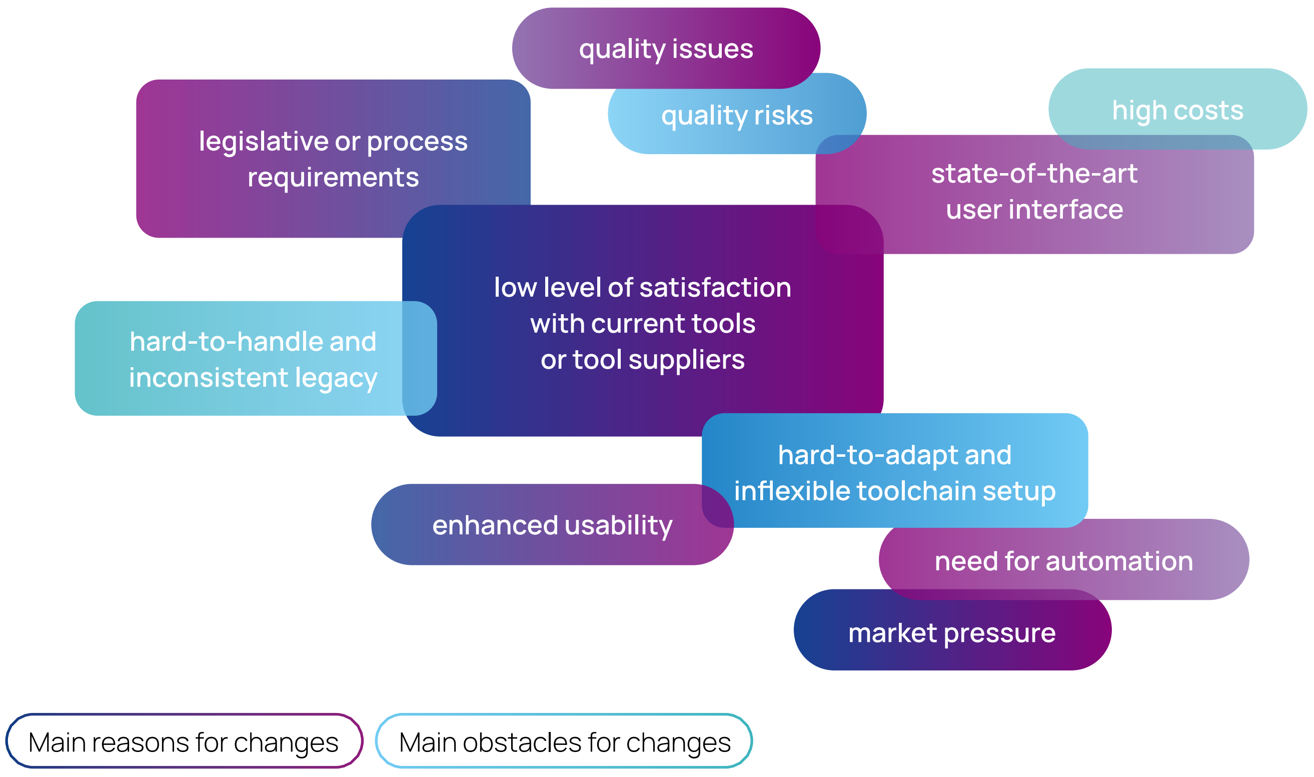
In the first instance, it is important to understand the triggers for the process as such, and to segregate the various steps within this process.

3.1 Reasons for ECU generation changes

Before taking a deep dive into the specific challenges, let's briefly look at the reasons why new ECU software must be developed in the first place and why this process is so time-consuming and complex. It all starts with a shift from one ECU generation to the next, entailing a substantial re-shaping of the software. Before programmers can even start with the creation of the new software, they must invest effort into the toolchain setup. Each new ECU generation has its own development process, depending on the specific domain and cooperation models. The corresponding toolchain is individually defined for this upcoming generation and can therefore contain considerable changes, in contrast to the minor adaptations that are solvable with updates. The guiding principle is to change as little as possible (or only as much as necessary) from one generation to the next, as a new toolchain always entails a high migration effort and project risks.

There are plenty of reasons why these changes from one generation to the next are needed (see figure 2). Some are intrinsic, such as quality issues or a low level of satisfaction with current tools or tool suppliers. Others are necessitated by external factors like new legislative or process requirements (e.g., ASPICE, ISO 26262 ASIL-D). The market pressure or an enhanced usability and state-of-the-art user interface also play a decisive role. The most common obstacles for a generation change consist in high costs, quality risks, and an inconsistent legacy, i.e. a hard-to-adapt, inflexible toolchain setup. A main need consists in a higher level of automation within the process.

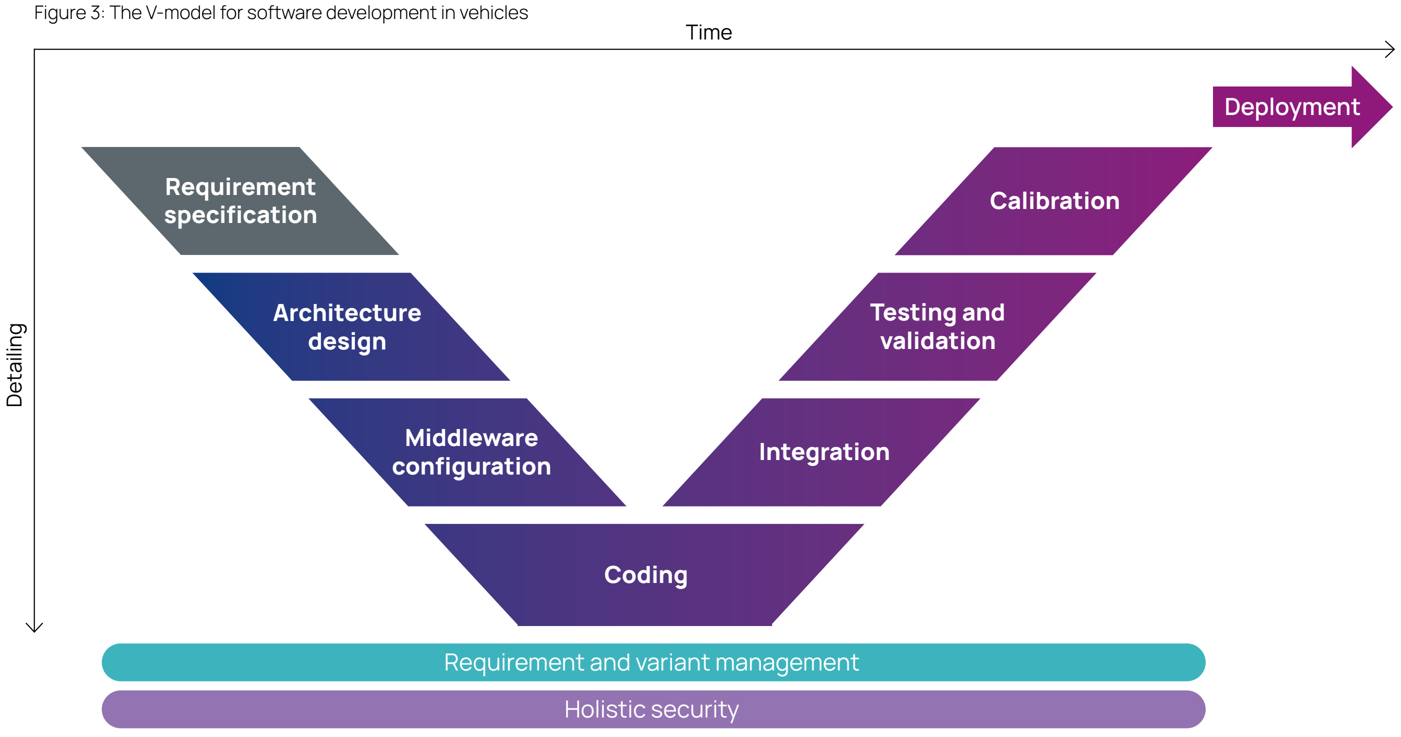
Figure 2: Reasons and obstacles for ECU generation changes


quality issues
quality risks
high costs
legislative or process requirements
state-of-the-art user interface
low level of satisfaction with current tools or tool suppliers
hard-to-handle and inconsistent legacy
hard-to-adapt and inflexible toolchain setup
enhanced usability
need for automation
market pressure
Main reasons for changes
Main obstacles for changes

3.2 Major steps in the development process

A common illustration of the development process for embedded software is the V-model, which comes in different variants. According to our experiences, we divided the phases into the following major work steps,as depicted in figure 3 below. They are not necessarily processed one after the other. Some can run in parallel or must be carried out several times. Requirement and variant management is necessary across all work steps and is iterated several times.

Figure 3: The V-model for software development in vehicles



The V-model splits software development into two major parts. This version is adapted to the ECU software development process.

Architecture design

As part of an ECU, a microcontroller is responsible for specific interlinked functionalities, such as engine or vehicle dynamics control. Hence, a very detailed requirements catalogue is key. The very first step consists in identifying all functionalities, constraints, hardware elements, the middleware, and their interdependencies. All hardware and software components must already comply with the rigorous automotive standards in this early phase, demanding meticulous attention to detail.

Middleware configuration

Middleware refers to the software layer between the operating system or hardware and the application software in an ECU. It serves as a bridge that facilitates communication and data exchange between various components within the vehicle's software architecture. By decoupling applications and the hardware-related systems, middleware makes it possible to develop, maintain, and upgrade them independently. Middleware acts as an intermediary, abstracting the underlying hardware and providing a standardized interface for software components for seamless interaction.

It goes without saying that such an intermediary must meet the highest security standards, as vulnerabilities and compliance issues in the ECU can lead to unauthorized access, data breaches, and system failures, compromising vehicle safety and reliability. Ensuring this level of compliance with stringent industry standards requires extensive testing and validation, adding complexity and time to the development process. The selection of middleware is therefore crucial, with a high focus on maturity (ISO 26262 ASIL-D compliance), future orientation (no vendor lock-in), and cybersecurity. Only a robust and highly configurable middleware solution with continuous updates can safeguard the ECU against evolving threats and fulfill all (changing) regulatory requirements.

Coding

Once the architecture has been defined and all components with their specific requirements are identified, the developers translate the system design into functional software. The required functionalities are already present in a standard process. Here, the software developers write new (supplementary) functions, eliminate errors, and optimize or extend existing functions. The challenge consists in setting up a process that ensures the possibility of implementing existing functions without needing to rewrite them. This implies high re-usability, while maintaining a high level of functional safety, cybersecurity, and code efficiency.

Integration

At this point, all the work streams can come together: the middleware configuration, information from the architecture design, the application software, as well as the pre-calibration data are built to a code that can be flashed onto the microcontroller.

A major challenge consists in optimizing, i.e. balancing the performance of the automotive function with the new hardware's capabilities to meet specific efficiency goals and environmental conditions. This involves fine-tuning the middleware for efficient system resource usage and real-time operation. Of course, safety and compliance are also crucial: meeting the stringent safety standards and regulatory requirements again means executing rigorous testing and certification processes.

Testing and validation

The testing and validation phase ensures that the functions of the ECU meet all requirements and specifications regarding safety, performance, and reliability. Extensive testing identifies any potential issues or malfunctions that may arise. This process can become extremely complex and lengthy in the case of software-heavy vehicles. Performing as many tests as possible within a short timeframe is therefore crucial to minimize costs and maximize performance.

Calibration

During calibration, the parameters and fields are filled with data, ensuring that the behavior of the software is adapted to the physical system. Some applications may consist of many thousands of parameters that influence each other. Also, parameters need to meet specific performance goals, environmental conditions, and regulatory standards. Adjustments must be implemented in later updates, e.g. when emission norms change.

Deployment

Once the software reaches the "ready for use" stage, it undergoes a final approval process and is fully flashed onto the target ECUs.

4. Main challenges of manufacturers and suppliers

Now that we have a good overview of the steps involved in the process, it is easier to match the specific challenges along the way. Some are bound to only one of the steps, some apply to the process as such or the shift from one generation to another. We have identified five major challenges that we encounter time and again throughout the automotive industry.

4.1 High integration effort

Vehicle systems are nearly always developed in a brownfield situation, i.e. in an already existing software environment, leading to a lot of legacy issues during integration and thus a complicated development process. The diverse and complex communication interfaces between various components necessitate intricate configurations and extensive testing to ensure seamless interoperability. Additionally, relying on manual processes for integration tasks leads to increased error rates, prolonged development cycles, and higher labor costs.

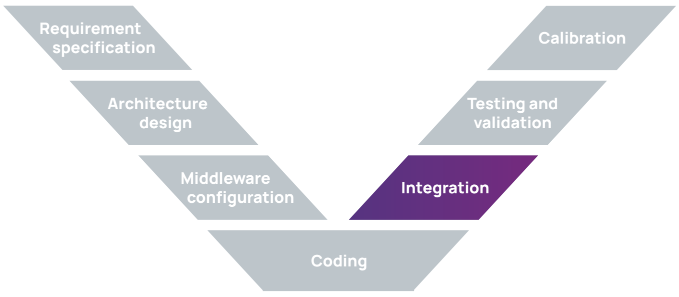


Figure 4: Allocation in the V-model within the process step “Integration”

Approaches to solving the challenge

Incremental modernization
Gradually update legacy components with modern equivalents, ensuring backward compatibility to facilitate a smooth transition and integration with new systems, and without completely overturning established processes or risking high investment costs.

Automation of integration tasks
Implement automated tools and scripts to handle repetitive integration tasks, reducing error rates and accelerating development cycles.

Middleware solutions
Employ state-of-the-art middleware to abstract and manage the complexity of interfaces, providing a consistent communication layer that simplifies integration efforts. Adopt and enforce standardized communication protocols to simplify interface configurations and ensure interoperability across different components.

4.2 Complex calibration

The time-consuming nature of calibration extends development cycles, pushes back project timelines, increases costs, and limits efficiency. Difficulties in understanding calibration parameters can lead to errors in the setup, which necessitate repeated testing and re-calibration cycles. Misconfigured parameters may result in the micro-controller failing to meet regulatory standards, entailing further revisions and potentially leading to costly recalls. Additionally, in the case of complex projects, conventional software documentation, which is sometimes even still performed manually, reaches its limits and is also highly error-prone.

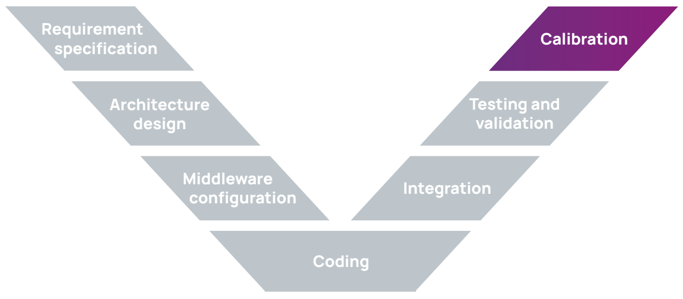
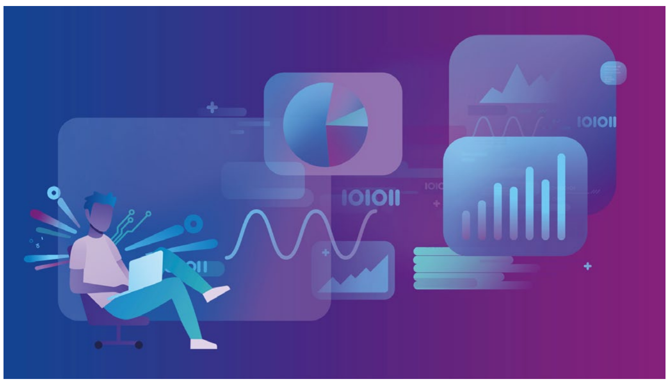


Figure 5: Allocation in the V-model within the process step “Calibration”



Approaches to solving the challenge

Enhanced documentation and knowledge sharing
Establish a comprehensive documentation that clearly explains each calibration parameter, its purpose, the interdependencies with other parameters, and its impact on the system. Implement automated documentation solutions to reduce errors and increase process efficiency.

User-friendly interfaces
Simplify the calibration process by using solutions that present only relevant parameters and options to the users. Interactive elements such as visual sliders and dynamic charts make it easier to understand and adjust calibration settings.

Automation and standardization
Use automated tools that can handle routine calibration tasks. They reduce the manual effort and streamline the calibration process, additionally ensuring consistency and reliability across different projects and teams.

Simulation and modeling
Utilize simulation and modeling tools to create virtual testing environments where calibration changes can be tested without negative consequences for the actual hardware. The impact of different calibration settings can be visualized for better decision making. Furthermore, model-based design approaches, where calibration parameters are embedded within system models, allow easier manipulation and testing.

4.3 Time-consuming testing and debugging

The more complex the software becomes, the more it needs to be tested – testing is becoming a bottleneck for innovation. Efficiency combined with reliability and safety is therefore a central element of a future-proof testing process. Additionally, the interaction between different modules and systems can lead to complex test scenarios that are difficult to manage and execute – and lead to a long integration process before testing even starts. On top of that, deeply embedded microcontrollers often have limited processing power, memory, and storage, which might constrain the types and scope of tests that can be run directly on the hardware. Hence, traditional debugging and testing tools may not be suited for the above-mentioned constraints of deeply embedded systems.

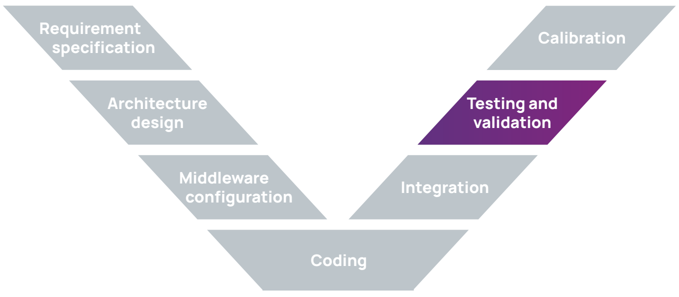
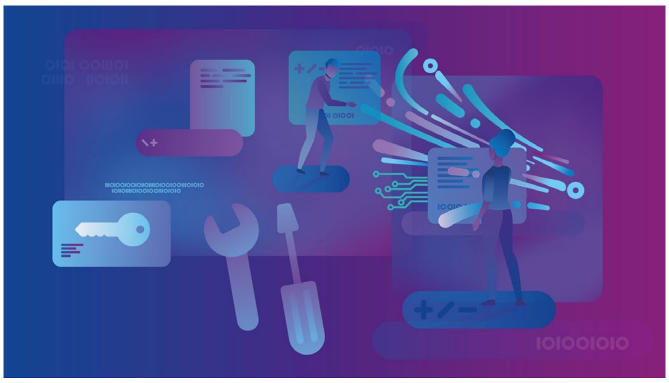


Figure 6: Allocation in the V-model within the process step "Testing and validation"



Approaches to solving the challenge

Model-based development
Use simulation tools to model and test systems in a virtual environment. This reduces hardware-based testing time as programming bugs are detected immediately and feedback is given to the programmer.

Advanced debugging tools
Use in-circuit emulators (ICE) for real-time debugging, as well as tracing and profiling tools to analyze execution flow, performance bottlenecks, and memory usage, aiding efficient debugging and optimization.

Software-in-the-Loop testing
Execute tests in a virtual environment (front-loading), which make it possible to detect errors at an early stage. Parallelization and time lapse also accelerate tests in the virtual environment.

Continuous integration/continuous deployment
Implement CI/CD pipelines to automate building, testing, and deployment processes, reducing manual effort and accelerating iteration loops. Automated tests detect issues early, cutting time and costs for bug fixes in development.

Unit testing and test-driven development
Adopt TDD practices to ensure that tests are done before coding and to achieve high test coverage. Develop and run unit tests for components before integration to simplify issue isolation and resolution.

Code generation
Automatically generate code from models, ensuring consistency and reducing the potential for human error.

Automation frameworks
Automate complex test scenarios with scripts to ensure consistency and reduce manual effort. Use code coverage tools to cover all execution paths and edge cases.

Parallel and continuous testing
Continuously test the system, whenever a change has been executed. Run the tests in parallel using multiple test rigs or virtual environments to reduce testing time. Optimize hardware resource allocation for efficient utilization and reduced bottlenecks.

4.4 Limited scalability and flexibility

Rigid, monolithic software designs restrict modifications and extensions, complicating updates and the addition of new features. Proprietary systems and a lack of standardization lead to vendor lock-in and complex, error-prone integrations with new functionalities or third-party components. Also, the insufficient modularity in software design prevents efficient code reuse and hinders the seamless integration of new features, affecting overall system adaptability.

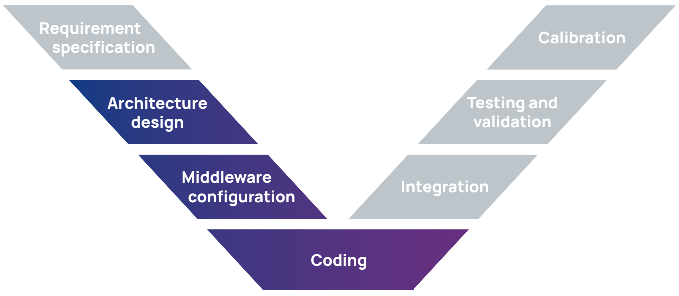
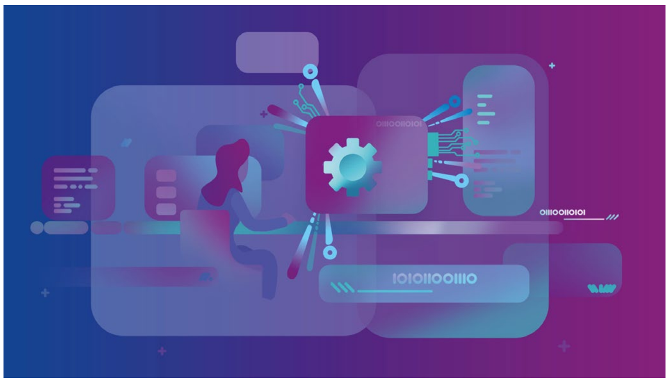


Figure 7: Allocation in the V-model across the process steps "Architecture design", "Middleware configuration", and "Coding"



Approaches to solving the challenge

Modular design and standardized interfaces
Implement a modular, component-based architecture and standardized communication protocols to break down the monolithic structure, allowing for easier updates and extensions.

Open standards and interoperability
Implement open standards and design systems systems into the toolchain with interoperability in mind to avoid vendor lock-in and simplify the integration of new functionalities or third-party components. Avoid suppliers who do not comply to common standards and thus provoke a vendor-lock in.

Middleware and abstraction layers
Utilize middleware and abstraction layers to decouple application logic among each other and from hardware specifics, ensuring easier updates and improved scalability.

Modularity and code reuse
Design software with a high degree of modularity and reusable components to facilitate the seamless integration of new features and improve overall system adaptability.

4.5 Holistic cybersecurity requirements

The diverse and evolving nature of cyber threats, including malware, hacking, and unauthorized access, complicates the task of securing ECU systems. Unfortunately, microcontrollers have limited computational resources, making it difficult to implement comprehensive security measures without impacting performance. Additionally, adhering to stringent and varying cybersecurity regulations and standards across different regions and industries adds complexity to the development process.

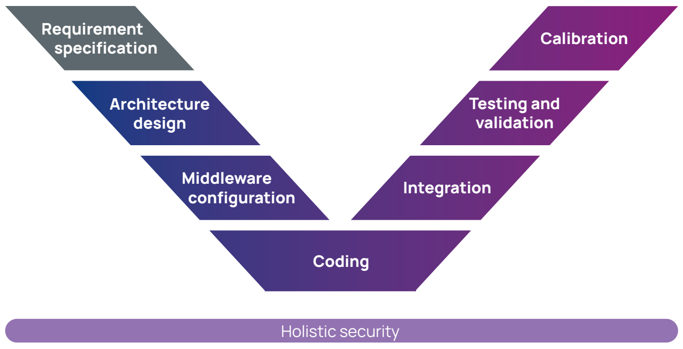
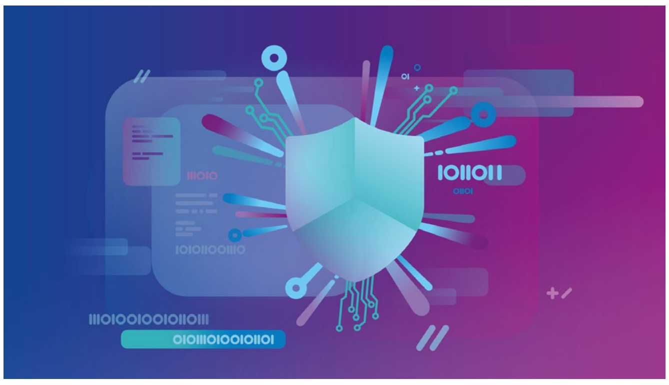


Figure 8: Allocation in the V-model across all process steps as a holistic challenge



Approaches to solving the challenge

Multi-layered security architecture
Address the diverse threat landscape comprehensively with a multi-layered security approach that includes encryption, authentication, intrusion detection, and secure boot mechanisms.

Security for resource-constrained environments
Use lightweight, efficient security protocols tailored to the limited computational resources of microcontrollers to ensure robust protection without compromising performance.

Alignment with regulatory standards
Ensure continuous monitoring and adapt to evolving cybersecurity regulations and standards; incorporate compliance requirements into the development lifecycle and conduct regular audits to ensure regulatory adherence.

5. Conclusion

Despite being the smallest component in the E/E architecture, micro-controllers are crucial in any setup – from function-specific control units to domain-independent zone ECUs. Investing in the improvement of microcontroller software development will remain an important factor for competitiveness in the automotive market over the coming decades. The aim is to increase speed, reduce costs, and guarantee security – all while taking legacy and increasing vehicle complexity into account. This can only be accomplished by reconsidering the established processes, including the gradual solution of the challenges presented in this white paper. Sophisticated solutions from experienced partners like ETAS allow OEMs and suppliers to optimize their deeply embedded software development, making it resilient against future (market) requirements and (legal) regulations. Such a strong basis creates long-term competitiveness in the global automotive industry.

About ETAS

Founded in 1994, ETAS GmbH is a wholly owned subsidiary of Robert Bosch GmbH with a local presence in all major automotive markets in Europe, North and South America, and Asia.

ETAS offers comprehensive solutions for the realization of software-defined vehicles in the areas of software development solutions, vehicle operating system, vehicle cloud services, data acquisition and processing solutions, integrated customer solutions and cybersecurity.

As industry pioneers in cybersecurity, we assist our customers in managing cybersecurity-related complexities, reducing cyber risks, and maximizing their business potentials with a proven on- and offboard portfolio of software products and professional security services.

ETAS automotive security solutions are safeguarding millions of vehicle systems around the world – and are setting standards for the cybersecurity of software-defined vehicles.

About our solutions

As a leading automotive supplier with more than 30 years of experience, we at ETAS have an extensive, proven portfolio of solutions. In the following, we have compiled a selection of bundles specially for ECU software development processes, tailored to your current needs and market requirements.

ETAS AUTOSAR solutions: easy, safe, and secure software integration

We offer OEMs and suppliers a solution bundle for the seamless integration of software and hardware, as well as comprehensive protection against cyber threats. Additionally, our virtual testing solutions enable early integration in the development lifecycle, resulting in reduced testing time and increased execution speed.

ETAS RTA-CAR (Classic AUTOSAR)
is a tool set used for developing and configuring Classic AUTOSAR software components, enabling efficient integration, code generation, and validation for ECUs
+
ESCRYPT CycurHSM
is an innovative and flexible HSM security firmware that ensures secure boot of the ECU, in-vehicle communication, ECU component protection, and flashing
+
ESCRYPT CycurLIB
is a cryptographic library specifically geared to the requirements of embedded systems
+
Virtual ECUs
to perform complex tests in early developmental stages

ETAS virtualized environment solutions: cloud-based software-in-the-loop testing

We provide modular tools and services for SIL testing in the cloud. They enable the creation, debugging, and pre-calibration of virtual ECUs as well as the integration and simulation of virtual artefacts, executed in parallel in the cloud. Engineers can test and iterate even large sets of test cases, resulting in a drastic increase in efficiency and simulation speed. Moreover, the cloud-based solutions are scalable and support continuous integration and deployment of software, which is particularly important for modern development processes.

ETAS VECU-BUILDER
is a tool to generate virtual ECUs for verification and validation of automotive microcontroller software in software-in-the-loop (SIL) setups
+
COSYM (CO-simulation of SYsteMs)
is the powerful simulation and integration platform to test and validate software at system level in the early development phase
+
ETAS MODEL-SYMULATOR
enables users to execute simulations in the cloud
+
ESCRYPT CycurFUZZ
is a state-of-the-art fuzz testing tool that helps meet current regulations and standards

ETAS calibration and documentation solutions: turning data into valuable information

Our seamless tool-coupling solution automates data search, analysis, and transmission. By integrating these proven products in one bundle, we boost the efficiency, accuracy, and team collaboration in ECU calibration and diagnostics.

ETAS EHANDBOOK
provides new tools for the documentation of ECU software, displaying the logic of the functions interactively and graphically at different levels of abstraction
+
INCA (Integrated Calibration and Application Tool)
contains flexible tools for the calibration, diagnostics, and validation of automotive electronic systems

Real-time calibrations validation solutions: unlocking data insights

Our solution bundle facilitates real-time validation of calibrations and rapid feedback, allowing for swift adjustments and high-quality outcomes. This integrated approach minimizes delays, enhances team collaboration, and accelerates development cycles, driving success in ECU projects and ensuring efficient project completion.

ETAS EHANDBOOK
gives an instant access to detailed ECU documentation
+
ETAS MDA (Measure Data Analyzer)
simplifies data analysis and visualization, uncovering trends and issues for precise calibration and optimized ECU performance
+
ETAS EATB (ETAS Analytics Toolbox)
automates and streamlines testing, providing powerful analysis and reporting tools for comprehensive insights and informed decision making
+
ETAS ASCMO (Advanced Simulation for Calibration, Modelling, and Optimization)
allows detailed simulation of ECU behavior and advanced calibration for higher performance and efficiency

Contact

Anthony Esteban
Customer Chief Engineer, ETAS
Get in touch on LinkedIn

Contact Form

ETAS GmbH
Borsigstraße 24, 70469 Stuttgart, Germany
T +49 711 3423-0, info@etas.com

Are you interested in
ETAS products or solutions?
Please visit **www.etas.com**

Or follow us on social media: