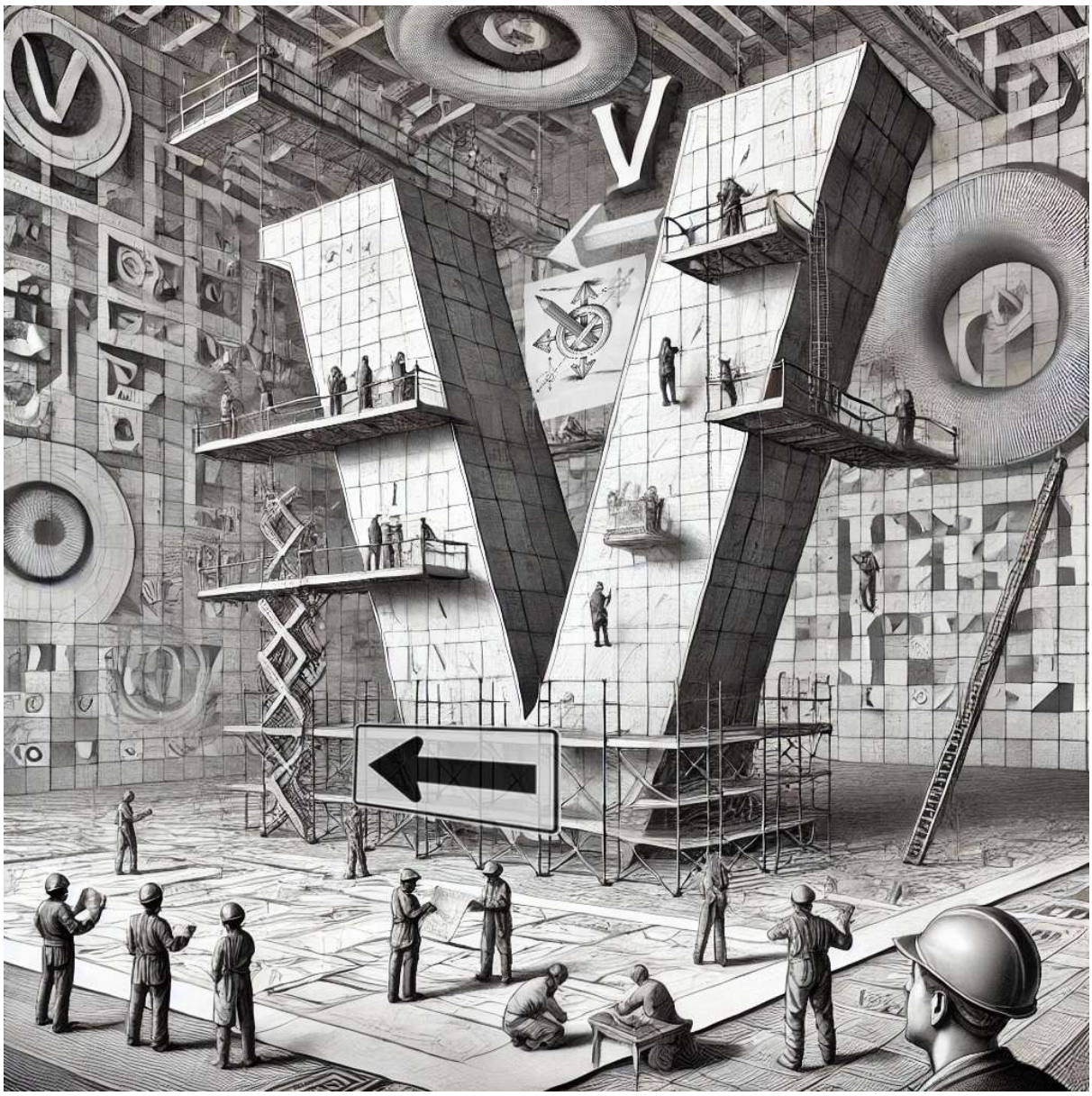
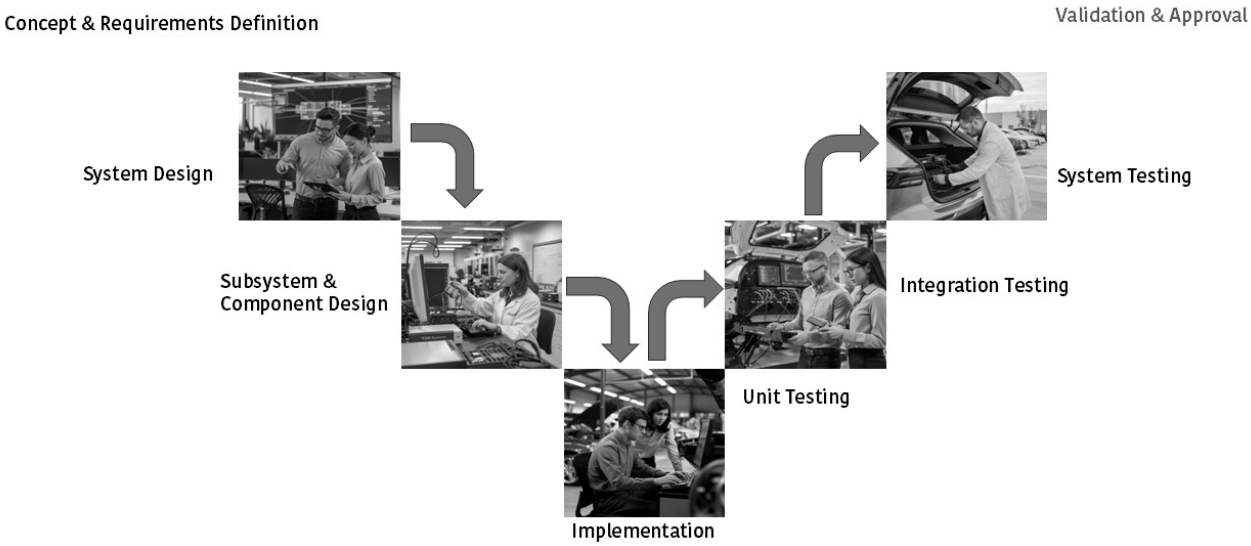
Shift Left in SDV: Testing Sooner, Smarter and Safer



The automotive industry is undergoing a profound transformation with the rise of Software-Defined Vehicles (SDVs), where software plays a central role in defining vehicle functionality, performance, and user experience. This shift demands a re-evaluation of traditional software development methodologies, particularly the V-Model, which has long been a cornerstone of automotive software engineering.

This article explores how the shift left approach enhances the V-Model not only for SDV development but also for validation in general and why it's a game-changer for the future of mobility.

Understanding the V-Model and its role in automotive development



Traditional V-Model

The V-Model is a sequential development process common in automotive engineering, emphasizing safety and reliability. It represents the software lifecycle in a "V" shape: The left side covers requirements decomposition into design and coding, while the right side covers integration and testing phases for verification and validation. In automotive contexts, the V-Model aligns each stage — from requirements analysis to system validation — with standards like ISO 26262.

However, the traditional V-Model has limitations when applied to SDVs. With vehicles now requiring tens of millions of lines of code to support features like advanced driver-assistance systems (ADAS), autonomous driving and over-the-air (OTA) updates, the sequential nature of the V-Model can lead to late-stage defect discovery, costly rework, and delayed timelines. This is where the shift left approach comes into play, adapting the V.

What Is the Shift Left approach?

The shift left approach involves moving critical activities — such as testing, validation, and quality assurance — earlier in the development lifecycle or "to the left" on the V-Model's timeline. Rather than waiting until the integration or system testing phases on the right side of the V, developers begin these activities during the requirements, design and coding stages. The goal is to identify and resolve issues as early as possible, reducing the risk of downstream problems and improving overall efficiency.

For SDVs, this approach is particularly valuable because of the complexity and interdependence of software components. By catching defects early, teams can avoid the exponential costs associated with fixing issues late in the process, especially in safety-critical systems where errors could have severe consequences.

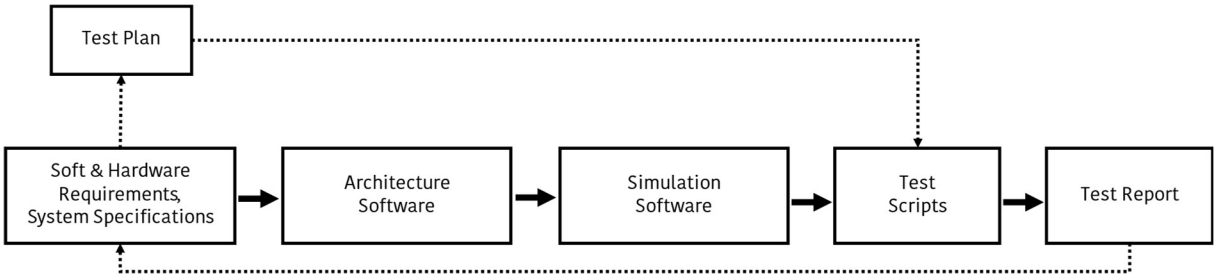
Applying Shift Left to the V-Model for SDVs

Integrating the shift left approach into the V-Model for SDV development requires a rethinking of traditional workflows. Here's how it transforms each phase:

1. Requirements Analysis and System Design

- In a shift left framework, testing begins at the requirements stage. Teams create executable models or simulations of requirements to validate their feasibility and correctness before moving forward. For SDVs, this might involve simulating how software will manage vehicle functions like battery optimization or sensor fusion for autonomous driving.

- Collaboration between developers, testers and domain experts ensures that requirements are testable and aligned with safety and performance goals from the outset.

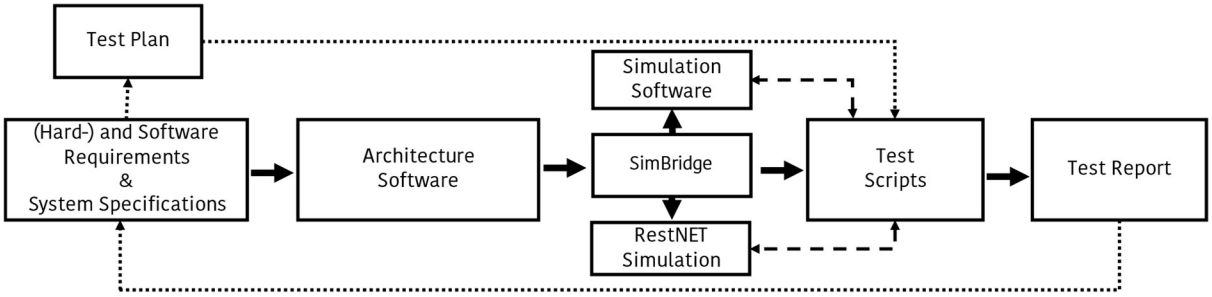


(I) *A validation toolchain needs to be set-up that covers the ability to simulate, create test scripts, and validate hard- and software of a new architecture.*

2. Detailed Design and Coding

- During the design and coding phases, shift left emphasizes unit testing and static code analysis. Developers write automated tests alongside code (e.g., using Test-Driven Development, or TDD) to verify individual modules in isolation.

- For SDVs, this could mean testing software components like an ECU's communication protocol or an ADAS algorithm before integrating them into the broader system. Early feedback loops help refine hardware specifications and software architecture, reducing the need for late-stage adjustments.

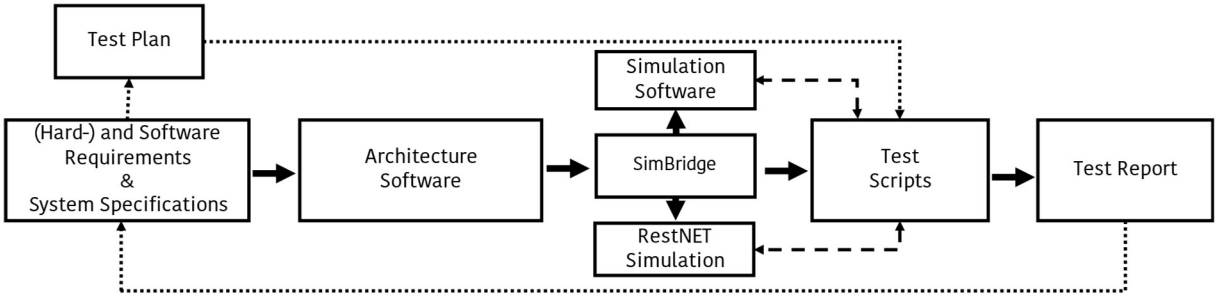


(II) *In addition to the above, the toolchain needs to add the ability to simulate the appropriate traffic, as well as necessary legacy devices "on the wire", so a new software component or algorithm could use this for its validation.*

3. Integration and testing

- While the right side of the V-Model traditionally focuses on integration and system testing, Shift Left ensures that much of this work is pre-empted. Virtual prototyping and software-in-the-loop (SiL) testing allow developers to simulate how components interact before physical hardware is available.

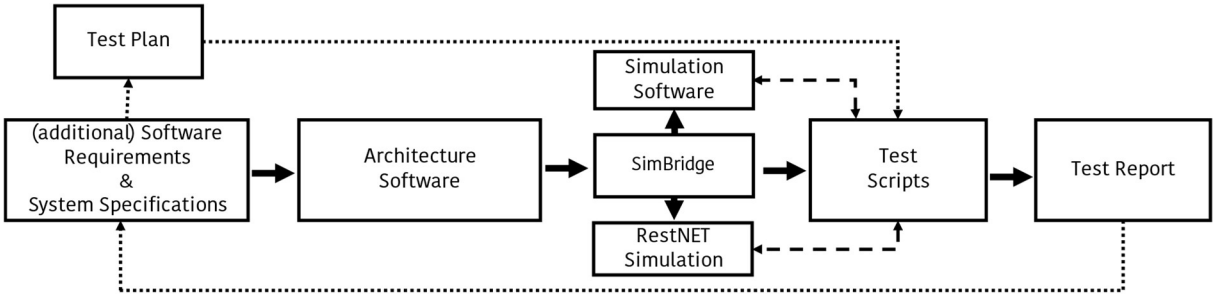
- In the SDV context, this is critical for validating complex interactions between software and hardware, such as a centralized control unit managing multiple vehicle zones. Automated regression testing further ensures that new updates don't introduce unintended issues.



(III) *In addition to the above, the toolchain needs to add the ability to integrate and/ or analyse the appropriate traffic including Inter-Zone Traffic on the "wire" so new components could be validated within the network.*

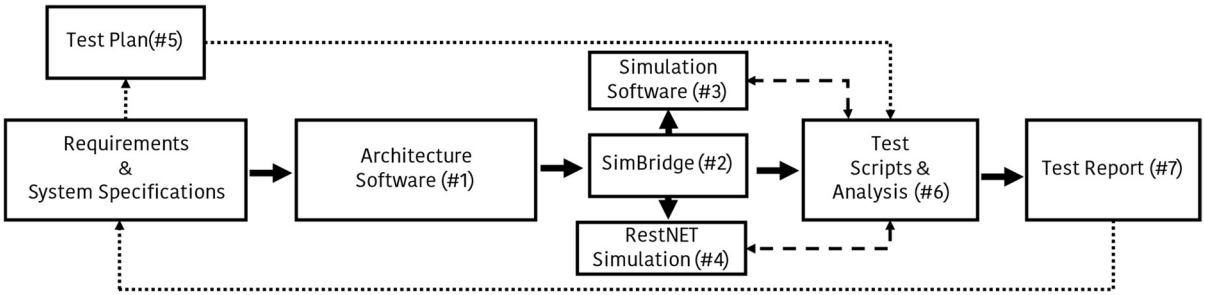
4. System Validation

- By the time system validation occurs, many defects have already been addressed, making this phase more about confirming overall performance than uncovering major flaws. For SDVs, this might involve vehicle-in-the-loop (ViL) testing or real-world simulations to ensure the software meets user expectations and regulatory standards.



(IV) *In addition to the above, the toolchain needs to add the ability to analyse the complete traffic including Inter-Zone Traffic and the whole system could be validated within the network. In addition, error injection (through scripted test scenarios) should provide further validation options.*

5. The Toolchain



A state-of-the-art toolchain includes, in addition to standard requirement management and modelling tools, a system architecture tool [see (#1)] such as PREEvision or Enterprise Architect. These software suites can export ARXML files, which are then passed to a SimBridge application (e.g. TSN System's SimBridge) [see (#2)].

The SimBridge is capable of reading and transforming ARXML data from the OEM's architecture tool into a format compatible with dedicated network simulation software [see (#3)].

This simulation software models communication networks to analyse time-critical performance, behaviour and reliability – without requiring physical hardware.

Additionally, the SimBridge can restructure and adapt data for use in test scripts, as well as prepare configuration data necessary for initializing real network hardware (e.g., network switches).

Finally, the SimBridge forwards processed simulation data (e.g., from OMNeT++) to a time-aware network analyser (e.g., TSN CoreSolution), which plays a central role in validation by executing the test scripts, evaluating the data, and generating test reports.

6. Conclusion

The shift-left approach in the automotive V-model involves moving development activities such as verification, validation, and testing to earlier stages of the lifecycle.

This fundamentally changes the required toolchain by necessitating tools that enable early simulation, continuous integration, automated testing, rapid prototyping, and early defect detection.

Traditional late-stage testing tools are complemented or replaced by tools that support virtual validation, software-in-the-loop (SIL), model-in-the-loop (MIL), and hardware-in-the-loop (HIL) simulations.

Consequently, the toolchain evolves from a sequential, stage-gated environment to an integrated, collaborative, and iterative ecosystem designed for rapid feedback, improved software quality, and reduced time-to-market.

Stage	Traditional V-Model	New V-Model with Left Shift
Concept & Requirements	Defined early, locked-in specs	Agile and iterative, flexible specs
System & Architecture Design	Fixed early in the process	Virtual models & software-first approach
Implementation	Late-stage software development	Software-driven, CI/CD integration
Testing & Validation	Sequential testing (late-stage)	Continuous testing (virtual & real)
Production & Deployment	Final release (hardware/software fixed)	Continuous software updates via OTA

This table shows the change in validation results from the old to the new V-Model.