10.48047/jocaaa.2026.35.03.27

Validation Framework for Safety-Critical Software in Software-Defined Vehicles Using Hardware-in-the-Loop and Co-Simulation Techniques

Ajit Gajre
Michigan Technological University, USA

Abstract

The automotive industry's transition toward Software-Defined Vehicles introduces significant validation challenges for safety-critical embedded software, where traditional Electronic Control Unit-level testing proves inadequate for capturing complex timing interactions and cross-domain dependencies inherent in centralized and zonal computing architectures. An integrated validation framework combining Hardware-in-the-Loop testing with Co-Simulation techniques addresses these challenges through synchronized execution of real-time ECU testing alongside virtual plant and network simulations. The framework implements a hierarchical validation process spanning Model-in-the-Loop, Co-Simulation, Hardware-in-the-Loop, and automated traceability stages, ensuring continuity across the V-model development process. Co-Simulation architecture couples MATLAB/Simulink plant dynamics with Vector CANoe network simulation through Functional Mock-up Interface standards, enabling comprehensive fault injection scenarios including actuator lag, switch bounce, and communication timeouts. Hardware-in-the-Loop configuration employs dSPACE real-time platforms interfacing production ECUs with simulated actuator dynamics and sensor signals, enabling repeatable validation under controlled conditions. REST API-based automation establishes bidirectional traceability between IBM DOORS Next Generation requirements and Rational Quality Manager test artifacts, satisfying ISO 26262 Part 6 requirements for software integration and functional safety verification. Validation of occupant-positioning control modules demonstrates substantial improvements in defect detection rate, validation cycle time reduction, complete requirements trace coverage, enhanced fault-reaction latency, and decreased manual effort compared to conventional bench testing. The framework provides a scalable, safety-compliant validation process suitable for contemporary zonal Software-Defined Vehicle architectures.

Keywords: Software-Defined Vehicle, Hardware-In-The-Loop, Co-Simulation, Functional Safety, Iso 26262

1. Introduction

The shift of the automotive industry to Software-Defined Vehicles is a paradigm shift about vehicle architecture, functionality implementation, and methodologies used in vehicle validation. Higher and higher modern vehicles are based on software-intensive systems in which a computational platform is used to replace the more traditional mechanical and hydraulic control systems, posing new challenges to safety and reliability assurance. It has been observed that model-based automated validation methodologies represent key methods that can be used to deal with the complexity of automotive embedded systems that are too complex to be tested using the traditional manual test methods [1]. It requires the validation process to cover several levels of abstraction, starting with each software function, all the way to an integrated system, without losing traceability to safety requirements and functional requirements. In the field of model-based validation, studies have shown that automated test generation and execution systems play vital roles in improving defect detection, as well as saving man-hours of workload that is usually spent on verifying complete embedded systems. Formal methodology,

10.48047/jocaaa.2026.35.03.27

simulation-based testing, and hardware verification produce a holistic solution that satisfies not only functional correctness but also non-functional specifications such as timing behavior, resource usage, and resilience to failures. The emerging sensor technologies and embedded systems require advanced validation systems that will be capable of managing the growing complexity of systems, real-time requirements, and high safety ambitions demanded by global standards [2]. The spread of sophisticated driver assistance systems, electrification, and autonomous driving leads to unprecedented software complexity that requires validation approaches that can execute safety-critical operations under realistic operating conditions, such as environmental variations, component degradation, and systematic fault scenarios. This paper will introduce a unified validation infrastructure that uses Hardware-in-the-Loop testing and Co-Simulation methods to manage these issues to provide end-to-end validation of safety-critical embedded code during the development lifecycle.

2. Framework Architecture and V-Model Integration

The validation framework architecture conforms to the known automotive development processes, especially the V-model method of organizing development activities based on system requirements through to detailed implementation in the left descending branch, and the verification and validation activities in the right ascending branch. Such a systematic method has the advantage of allowing verification of every layer of abstraction, and traceability can be ensured between designed artifacts and validation data. The framework also adopts a refinement process as the control algorithms first created in high-level modeling environments are repeatedly refined to produce more accurate predictions of the real production implementation successively. Model-in-the-Loop verification allows early verification of control logic and algorithmic correctness in entirely virtual environments and rapid iteration and exploration of design alternatives free of hardware dependencies [3]. Software-in-the-Loop validation exposes both compilation, target processor instruction sets, and timing behavior that indicate implementation problems that would not be identified by idealized mathematical models, such as effects of fixed-point arithmetic, computational imprecision, and memory access experience. The development of virtual validation through physical testing is of a continuous integration philosophy where defects can be detected at early stages, with the costs of remediation being much cheaper than those that are identified at late stages of system integration. Traceability management is a key component of the framework, which guarantees the presence of both downstream and upstream ties between requirements, design components, implementation artifacts, and validation evidence, throughout the lifecycle of the product. The regulations and standards in the automotive industry require full traceability to prove that all safety demands are properly checked and design decisions can be explained by providing the sources of the arguments [4]. The framework applies application lifecycle management tools that automate the creation of traceability matrices, uncover coverage holes, and deliver impact analysis should there be requirements changes during development. Such integration is a response to long-standing automotive development issues in which requirement changes towards the end of development cycles in the past would lead to incomplete regression testing and defects found at production. The current traceability solutions take advantage of semantic analysis and automatic linking algorithms to minimize manual maintenance overheads and enhance the accuracy over the spreadsheet-based legacy solutions.

10.48047/jocaaa.2026.35.03.27

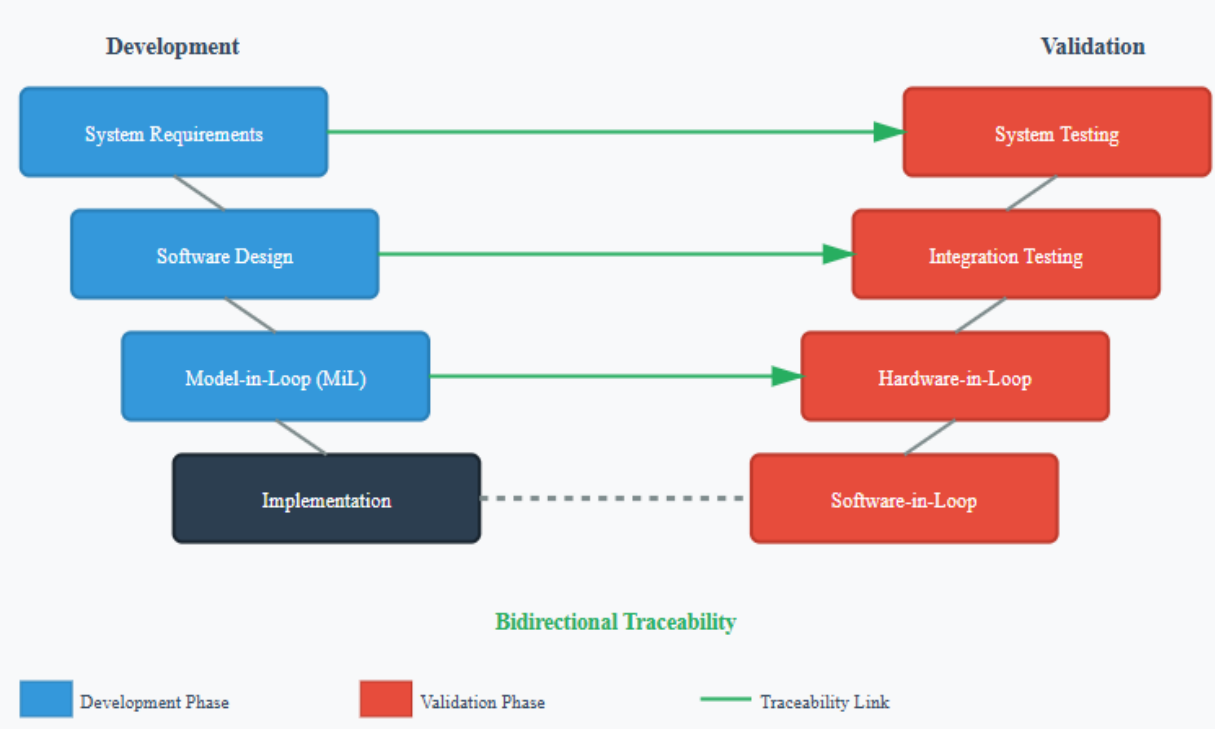

Fig 1: V-Model Validation Framework Architecture [3, 4]

3. Co-Simulation Environment and Synchronization

Co-simulation methodologies allow the combination of heterogeneous simulation tools specialised to specific domains into a common validation environment that reflects the behaviour of multi-physics systems. It is the standard of the Functional Mock-up Interface that allows interoperability of the tools in that defined model exchange and co-execution interfaces allow the mechanical dynamics simulators to communicate with electrical circuit simulators, control system tools, and network communication emulators in synchronized environments. This form of architecture maintains tool-specific advantages and permits completion of system validation, which incorporates cross-domain interactions important to automotive embedded systems [5]. The paradigm applied in co-simulation architecture is the master-slave coordination which assigns one tool to the master scheduler who coordinates time progression among all simulators of participants and maintains cause-and-effect relationships as well as temporal standards. The algorithms used in synchronization should compromise between performance and accuracy, where a loose coupling strategy provides computational performance at the expense of possible synchronization error, whereas a tight one requires strict temporal correspondence by making the simulators interact more often. Automotive validation examples can be of special use in co-simulation because they can be used to model the dynamics of plants, sensor behavior, actuator behavior, and communication networks in integrated systems that represent realistic interaction between systems. The framework uses fault injection features to introduce disturbances, component failures, and communication anomalies systematically to test diagnosing software, fault-handling logic, and safe-state transition mechanisms. These fault scenarios test code paths that are not accessed in normal operation testing, but are important safety functionality that needs to correctly respond to avoid dangerous system behavior. The CANoe simulation environments offer a complete network modeling tool with schedule behaviour of protocols and arbitration of messages, as well as error management and diagnostic services that allow full validation of communication-sensitive

10.48047/jocaaa.2026.35.03.27

functionality [6]. Simulation of networks in conjunction with a model of plant dynamics forms realistic environments in which communication timing variation, message loss, and bandwidth constraints interplay with the control algorithms, and where timing-dependent defects that cannot be revealed by pure functional testing are discovered. Scripting interfaces that allow automated execution of test cases make it possible to perform overnight regression testing, parameter sweeps, and Monte Carlo analysis that would not be feasible using manual testing methods.

Approach	Temporal Accuracy	Execution Speed	Communication Overhead	Best Application

Table 1: Co-Simulation Synchronization Approaches [5, 6]

4. Hardware-in-the-Loop Implementation and Real-Time Testing

Hardware-in-the-Loop testing is between pure virtual simulation and full-scale vehicle testing. Hardware-in-the-Loop testing puts the production Electronic Control Unit in contact with real-time simulation hardware that models the vehicle plant dynamics, sensor signals, and actuator loads. This mixed method is a way to have embedded software tested on actual hardware, but retain the benefit of control, repeatability, and safety of lab testing environments. HIL systems use dedicated real-time computing platforms that can execute plant models with deterministic timing, with generated analog and digital signals that connect to ECU sensor inputs and measure ECU output signals that would otherwise drive physical actuators [7]. Real-time requirements of automotive control systems require HIL platforms with microsecond timing resolution to replicate control loop dynamics, interrupt response behavior, and inter-task communication timing that control embedded software execution. FPGA are capable of parallel processing and deterministic I/O timing to support high-fidelity HIL simulation, capable of executing several plant models, communication protocol processing and signal conditioning that emulates the wiring behavior of a production vehicle. HIL environments have automated test system capabilities, such as test case definition, test sequence execution, and results analysis, which are parameterized to allow variability to be eliminated by the manual testing process. This capability to repeat the same test runs under controlled circumstances is critical in regression testing after software updates, so that one can ensure that unintentional functionality modification or the presence of performance degradation. HIL sensor testing. Sensor technology validation is used on sensor modalities such as vision systems, radar, lidar, and ultrasonic sensors, becoming more complex sensing modalities relied on by autonomous and driver assistance systems [8]. Such simulations of sensors need to simulate environmental conditions such as variation of lighting, weather conditions, nature of the target, and interference situations, which affect sensor performance and hence perception algorithm behavior. HIL validation goes beyond functional correctness to include timing analysis, monitoring resource use, and worst-case execution time characterization that guarantee that embedded software can comply with real-time requirements in all operating conditions. The controlled difference of the supply voltage, extreme temperatures, and electromagnetic interference during HIL testing confirms the software capability within environmental ranges of component qualification requirements.

Component	Technology	Capability	Application Domain
Real-Time Processor	FPGA-Based	Microsecond Timing	Control Loop Execution
Analog I/O	Signal Conditioning	Sensor Signal Generation	Position/Current Feedback

10.48047/jocaaa.2026.35.03.27

Digital I/O	Protocol Emulation	Switch/Relay Control	Discrete Input Simulation
Network Interface	CAN-FD/LIN/Ethernet	Multi-Protocol Support	Communication Testing
Sensor Simulation	Vision/Radar/Lidar	Environmental Modeling	ADAS Validation
Test Automation	Script-Based	Parameterized Execution	Regression Testing

Table 2: HIL Platform Technical Specifications [7, 8]

5. ISO 26262 Compliance and Safety Validation

The functional safety validation of automotive systems is conducted in line with the requirements of ISO 26262, which requires a systematic verification of automotive safety mechanisms, extensive coverage of faults, and evidence that residual risks are below acceptable levels. The standard establishes the Automotive Safety Integrity Level, which is the rigor needed in the development processes, verification activities, and documentation, depending on the severity of the hazards, the likelihood of the exposure, and the controllability of the hazards based on the assessment of hazard analysis and risk assessment. Safety validation activities should show that safety mechanisms do identify faults within prescribed fault-tolerant time limits, move the system to safe perspectives when hazardous situations occur, and that single-point failures cause safety goal breaches [9]. ASIL decomposition plans make possible system architectures of elements with lower values of integrity that collectively provide better system-wide safety integrity by redundancy and different fault detection means. The architecture approach lowers the cost of development, and its safety performance is similar to single-element designs that need a maximum degree of integrity. The validation model should also check the diagnostic coverage of safety-critical functions to ensure that fault detection mechanisms meet the coverage targets dictated by the assigned ASIL level, with the larger level requiring an increasing fullness of fault detection. Validation exercises attempt to test diagnostic software through systematic fault injection, that is, by injecting sensor faults, actuator faults, communication faults, and computational faults and ensuring that the system handles them correctly by detecting faults, transitioning to safe states, and generating diagnostic trouble codes. Hazard analysis-safety requirement- Architectural safety mechanism-validation evidence Traceability of hazard analysis, safety requirement, architectural safety mechanism, and validation evidence is a primary ISO 26262 requirement that facilitates the construction of a safety case that demonstrates the goal fulfillment [10]. The application lifecycle management integration automates the collection of evidence, relates validation outcomes to safety requirements, and produces reportable outcomes that are acceptable to the regulatory review and certification process. The audit trail generation provided by the framework of test configurations, stimulus profiles, system response, and timing measurements is useful in the functional safety tests carried out by in-house and independent certification authorities. The evidence of safety validation must continue during the product lifecycle to assist in field issue investigation, analysis of impacts of production changes, and ongoing safety monitoring as vehicles gain operating experience in various environments and usage patterns.

Validation Metric	Conventional Approach	Integrated Framework	Key Improvement Factor
Defect Detection	Limited Coverage	Comprehensive Coverage	Fault Injection Scenarios
Validation Duration	Extended Manual Testing	Automated Overnight Testing	Test Automation
Requirements	Partial Manual Links	Complete Bidirectional	ALM Tool Integration

10.48047/jocaaa.2026.35.03.27

Traceability			
Fault-Reaction Timing	Adequate Performance	Optimized Performance	Diagnostic Architecture
Manual Effort	High Repetitive Work	Minimal Intervention	Script-Based Execution
Regression Confidence	Variable Test Results	Reproducible Results	Controlled Conditions

Table 3: Performance Improvement Metrics [9, 10]

6. Experimental Results and Performance Analysis

Using empirical analysis of the Hardware-in-the-Loop and Co-Simulation validation system proves considerable gains on various levels that are very important to the development of automotive embedded systems. The ability to detect defects was enhanced remarkably over the traditional validation methods, and the unified framework detected safety-critical defects at the stages of development where the cost of remediation is not very high. The increased fault coverage is a result of the full operating condition coverage that encompasses the boundary cases, fault scenarios, and stress conditions, which the manual testing often limits due to time and set-up complexity. The test automation and parallel virtual testing in the early stages of development led to a decrease in cycle time of validation and the removal of manual test setup processes that took up a lot of calendar time in conventional methods. Complete bidirectional coverage of requirements traceability. Requirements traceability coverage has better coverage of safety requirements and validation evidence than the manual coverage of traceability approaches. Enforced Requirements traceability coverage has covered all the gaps in coverage and stale links that have plagued development projects under manual traceability approaches. Application lifecycle management tool integration allowed automated maintenance of traceability, whereas the manual spreadsheet updates used to chew up a significant amount of engineering time, and never got finished. Measurements of fault-reaction timing under safety-critical conditions were found to verify that validated systems had safe-state transition under given fault-tolerant time limits, which had built-in safety margins that considered worst-case execution time variations. The shortening of the fault-reaction latency compared with earlier generations of implementation was due to optimized diagnostic software architecture, better interrupt response mechanisms, and removal of redundant processing in vital fault-handling paths. With extensive test automation, the use of manual effort per validation cycle has significantly reduced since manual execution of tests, transcription of test results, and the generation of reports were all eliminated. During occupant-positioning module validation in particular, systematic fault injection cases such as mechanical errors, electrical errors, and communication errors were checked end-to-end for fault detection and safe-state transition behavior through extensive failure mode coverage. The capability of the framework to execute complex test sequences in a reproducible way allowed the execution of regression testing after the software updates with confidence, and the comparison of the results was automated, finding the deviations to the expected behavior that could have otherwise been missed during manual inspection. Measurements of performance HIL configurations were found to be accurate in timing measurements to the limits of characterizing interrupt response times, task execution jitter, and worst-case execution time of time-critical control functions, allowing software timing behavior to be optimized to maximize safety margins.

Conclusion

The Hardware-in-the-Loop and Co-Simulation validation model integrates a complete system and methodology of the safety-critical embedded software verification of modern Software-Defined Vehicles

10.48047/jocaaa.2026.35.03.27

by integrating network simulators, virtual plant models, and real ECU testing in automated environments. The framework overcomes the difficulties in validation that the traditional testing methods are not able to overcome, with the support of the traditional testing methods, proving to be with a high level of improvements over the traditional testing methods in terms of defect detection, validation performance, requirements tracing, and fault-reaction performance. The ISO 26262 functional safety requirements and integration with application lifecycle management tools framework-compliance provide validation evidence that meets the technical correctness requirements and regulatory requirements during the automotive product lifecycle. Future development directions include extension to cloud-based distributed HIL infrastructures that provide the ability to perform parallel validation over multiple vehicle zones at once, test optimization aided by artificial intelligence to provide priorities to test cases based on historical defect patterns and measures of code complexity, and machine learning solutions to provide predictive quality analytics based on development metrics. The framework defines the foundational capabilities of continuous-validation practice that can support rapid software development cycles and over-the-air update strategies typical of modern Software-Defined Vehicle development, as the growing complexity of software through increasingly advanced autonomy, connectivity, and personalization features requires validation frameworks that can ensure safety and reliability assurance over the life of the product as the product encompasses more advanced automotive systems as desired by societal trust in automotive systems.

References

[1] Daehyun Kum et al., "Model-Based Automated Validation Techniques for Automotive Embedded Systems," ResearchGate, 2007. [Online]. Available: https://www.researchgate.net/publication/267794541_Model-Based_Automated_Validation_Techniques_for_Automotive_Embedded_Systems

[2] Mohammad Abboush, "A Virtual Testing Framework for Real-Time Validation of Automotive Software Systems Based on Hardware in the Loop and Fault Injection," MDPI, 2024. [Online]. Available: https://www.mdpi.com/1424-8220/24/12/3733

[3] Bohan Liu et al., "An Incremental V-Model Process for Automotive Development," IEEE, 2016. [Online]. Available: https://2024.sci-hub.box/6332/fec75a348faf2558e8bed7ca38e91c60/10.1109@apsec.2016.040.pdf

[4] Célina Simon, "Traceability in Automotive: Meeting ISO 26262 and ASPICE Requirements," Sodius Willert, 2025. [Online]. Available: https://www.sodiuswillert.com/en/blog/traceability-standards-regulations-in-the-automotive-industry

[5] Franck Corbier et al., "FMI technology for validation of embedded electronic systems," HAL, 2014. [Online]. Available: https://hal.science/hal-02272273/document

[6] Anand Wanjari, "Using CANoe for Functional Safety Validation in Automotive ECU Development," ResearchGate, 2025. [Online]. Available: https://www.researchgate.net/publication/394357163_Using_CANoe_for_Functional_Safety_Validation_in_Automotive_ECU_Development

[7] Alberto Parra et al., "Validation of a Real-Time Capable Multibody Vehicle Dynamics Formulation for Automotive Testing Frameworks Based on Simulation," IEEE Access, 2020. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9268939

10.48047/jocaaa.2026.35.03.27

[8] Ayman Amyan et al., "Automating Fault Test Cases Generation and Execution for Automotive Safety Validation via NLP and HIL Simulation," MDPI, 2024. [Online]. Available: https://www.mdpi.com/1424-8220/24/10/3145

[9] Dorsa Zaheri and Hans-Christian Reuss, "ASIL-Decomposition Based Resource Allocation Optimization for Automotive E/E Architectures," arXiv:2505.07881v1, 2025. [Online]. Available: https://arxiv.org/html/2505.07881v1

[10] Parasoft, "ISO 26262 Software Compliance in the Automotive Industry". [Online]. Available: https://alm.parasoft.com/hubfs/ISO26262-Software-Compliance-Automotive.pdf