Reducing Rework in Automotive Software through Simulation-Guided Verification

Larissa Pestana
Stellantis
Recife, Brazil
larissa.pestana@stellantis.com

Breno Miranda
Federal University of Pernambuco
Recife, Brazil
bafm@cin.ufpe.br

ABSTRACT

The development of automotive software predominantly follows the V-cycle, where each development phase has a corresponding verification phase. However, integration verification, which considers system complexity, only occurs at the end of the cycle. This results in a disparity between software verification and system verification, where functionally validated code during development becomes dysfunctional during system verification. The lack of adequate preparation leads to significant rework, with algorithm errors being identified only at the end of the cycle, delaying the transition from development to the initial phase of the V-cycle and increasing both costs and time. This research initially explores multiple verification approaches. After analyzing their relevance to the addressed issue, the focus is placed on Simulation-Guided Verification, which has the potential to transform the V-cycle into a W-cycle or V-INC (Verification Incremental Cycle), by inserting in-loop verifications at each development phase. Techniques such as Model-in-the-Loop (MiL), Software-in-the-Loop (SiL), Processor-in-the-Loop (PiL), Hardware-in-the-Loop (HiL), and Virtual-in-the-Loop (ViL) are employed for continuous and iterative verifications, ensuring early fault detection and significantly reducing rework, thereby enhancing the efficiency of automotive software verification.

KEYWORDS

Automotive Software Verification, Simulation-Guided Verification, V-Cycle, Integration Verification, In-Loop Verification Techniques (MiL, SiL, PiL, HiL, ViL).

1 INTRODUCTION

The automotive industry's increasing complexity of embedded systems necessitates revised verification and validation processes. Rigorous validation stages are essential to ensure quality, safety, and functionality in critical systems, from propulsion to advanced driver assistance. Traditionally, the V-cycle is used, but its linear approach is rigid, slow to adapt to changes, has long feedback cycles, and is inefficient for complex systems. These limitations result in high costs and increased risk of failures during final verification.

To overcome the V-cycle's limitations, Liu *et al.* [5] proposed the W-cycle and V-INC (Verification Incremental Cycle). The W-cycle includes early verification and validation, allowing early defect detection and reducing correction costs. A system model is built and tested before implementation, enabling quicker corrections. The V-INC cycle is more iterative and incremental, continuously developing, refining, and integrating models and test cases, with simulations running throughout the lifecycle to verify the system virtually and avoid delays until integration and system testing phases.

The increasing challenges and costs of automotive software development have prompted studies to identify critical points needing attention. Falcini and Lami [2] conducted an empirical study on system and software testing practices using the Automotive SPICE methodology, as described by VDA QMC Working Group 13 / Automotive SIG [9], a process assessment and improvement model specific to the automotive industry.

Results indicated that software unit verification (SWE.4) and software integration verification (SWE.5) often have lower performance ratings due to informal verification activities and poorly defined testing strategies. The research also highlighted a significant disparity in quality between system testing and software testing, with system testing being more mature. Research on development cycle methodologies (V-cycle, W-cycle, V-INC) and critical software testing areas shows inefficiencies in software verification during development and integration. Algorithmic errors are often identified only during final system verification, leading to significant rework. This underscores the need to improve early-stage verification and validation practices, a focus of this research.

2 VERIFICATION CYCLE APPROACHES

2.1 Static Code Analyzer

The static code analyzer is the most common tool in automotive software verification, identifying errors like variable type mismatches. However, it cannot verify properties dependent on the software's dynamic behavior in real environments. While detecting syntactic and logical errors, it doesn't guarantee full system functionality under real conditions. This research aims to ensure the reliability of automotive software in complex, dynamic scenarios.

2.2 Formal Methods

Formal Methods are rigorous mathematical techniques used to verify specific properties of computational systems, ensuring their correct and safe behavior. TORODOV *et al.*[8] explore the application of three main Formal Methods techniques in the context of automotive software: abstract interpretation, model checking, and deductive proofs.

2.2.1 Abstract interpretation. identifies runtime errors, such as division by zero and out-of-bounds array access, by computing abstract representations of the code without the need for actual execution.

2.2.2 Model checking. verifies whether a formal model of the system satisfies specific properties expressed in temporal logic. This method is effective in identifying failures through counterexamples, which show paths that violate the verified properties.

2.2.3 Deductive proofs. establish mathematical properties of formal models, offering a more expressive approach. However, they require a high level of expertise and do not provide direct counterexamples when a property is violated.

Formal Methods detect errors more efficiently than traditional testing and should be gradually incorporated. While they provide high confidence in system correctness, they face scalability challenges and barriers like the learning curve for engineers and implementation costs in the automotive sector.

2.3 Simulation-Guided

Simulation-guided verification ensures the correctness and reliability of automotive control systems, particularly powertrain controls, by combining simulation and optimization to identify undesirable behaviors and meet specified requirements. Accurate models of the system and its environment are created and simulated under various conditions to estimate behavior. Kapinski *et al.* [4] discuss using simulations to validate functional behavior, adjust control parameters, and estimate performance.

Verification is iterative: a simulation engine generates behaviors, and an optimizer searches for inputs and parameters that may cause failures. This can be done using Open-Loop and Closed-Loop approaches.

2.3.1 Open-Loop. The system model is tested in isolation, without feedback from the environment or the controlled plant. In this context, the plant is represented by the vehicle. The system inputs are provided without considering the system's responses over time. The objective of open-loop testing is to validate whether the controller, in this case, the Electronic Control Unit (ECU), meets the basic functional requirements under predefined conditions.

2.3.2 Closed-Loop. the entire system is simulated, including the controller (such as the Electronic Control Unit - ECU) and the plant (the vehicle itself), allowing for dynamic feedback between them. In this context, the controller, which is the ECU, sends commands to the vehicle system (the plant), and the resulting feedback from the vehicle's behavior is used to continuously adjust the controller's output. This replicates real operating conditions of the system, where the controller dynamically monitors and adjusts the plant's response to its inputs. This method offers a more accurate representation of the system's behavior under real operating conditions. The closed-loop testing configurations include: Model-In-The-Loop (MIL), Software-In-The-Loop (SIL), Processor-In-The-Loop (PIL), Hardware-In-The-Loop (HIL).

Closed-loop is the approach that fits the requirements of this study, which seeks improvements for software verification before integration with simulations closer to system verification.

2.4 Simulation-Guided in Closed-Loop

2.4.1 Model-In-The-Loop (MiL). MiL is an initial approach in the verification process where both the controller and the plant are represented by software models. This allows engineers to test the control system design in a detailed virtual environment before implementing it in real hardware. MiL simulations are executed on a host PC, offering flexibility to quickly modify and adjust the models. This approach enables the early identification of design errors, providing detailed and flexible validation of the controller in a safe and controlled environment. The main advantages of MiL include the ability to test multiple operational scenarios and the ease of integration with other software development tools, such as Simulink.

2.4.2 Software-In-The-Loop (SiL). SiL involves implementing the controller with production code while modeling the plant in software, testing the actual code in a simulated environment. This validates controller logic and identifies implementation issues before hardware integration. The main advantage is ensuring the controller meets performance and functionality requirements.

Jeong *et al.* [3] conducted a case study using SiL for initial testing of complex automotive systems, employing the AUTOSAR methodology to configure and model software components and map them to electronic control units. The virtual functional bus layer simulates communication and interaction between components using shared memory and an event manager. To integrate SiL simulation with a vehicle simulator, the authors connected the code simulator to the vehicle simulator, allowing testing of control functions using the vehicle's sensors and actuators, reproducing the physical effects of automotive software actions in a virtual environment.

2.4.3 Processor-In-The-Loop (PiL). PiL involves executing production code on target hardware (e.g., the vehicle's ECU) while the plant is simulated on a host PC. The controller and plant communicate via a direct link, such as Ethernet or CAN bus. PiL tests the interaction between production code and control hardware in a near-real environment, identifying issues not evident in earlier MiL and SiL phases. Its main advantage is validating production code performance on target hardware in real-time, ensuring correct system function under real conditions.

Muttenthaler *et al.* [6] propose automatic verification of hardware/software integration in microcontroller environments. Tests are executed on the host, interacting with the HIL system and communicating with the microcontroller via a debugger, eliminating extensive instrumentation. The methodology enables functional verification of software components using a PiL setup, automating hardware/software integration verification. A test platform verified low-level driver software components in their microcontroller environment. Results showed automated testing was significantly faster than manual testing, reducing costs and increasing verification efficiency.

2.4.4 Hardware-In-The-Loop (HiL). HiL is the most advanced and realistic approach in the verification cycle. In this configuration, both the controller and the plant are represented in hardware. The controller receives electronic inputs from the virtual plant (simulated in hardware) and sends outputs to it. This creates a highly realistic test environment that simulates the behavior of the entire system under conditions nearly identical to those found in a real vehicle. HiL is essential for system verification, the final validation of the system before implementation in a real vehicle, as it allows testing the performance of the controller and the plant under dynamic and complex operational conditions. The main advantages of HiL include the ability to conduct extensive and realistic tests without the risks and costs associated with real vehicle testing.

2.5 Special Loop: Virtual Verification

The need for real systems for testing throughout the development cycle has led the automotive industry to research Virtual HiL, which simulates HiL for system verification. Researchers from General Motors and universities conducted a feasibility study on the Automotive Virtual Verification Ecosystem. Ågren *et al.* [1] identified significant benefits of virtual verification, including early software integration tests, early error detection, and increased test repeatability. However, adoption faces challenges such as the lack of high-fidelity modeling technology and a global strategy.

Collaboration between industry and academia is necessary to overcome these challenges. Advances in virtual verification research are essential. A significant advance was made through collaboration between Mentor, a Siemens company, and Ain-Shams University. Safar *et al.* [7] discusses an integrated framework for virtual verification and validation (VVV) of complete automotive systems. The proposed framework simulates the system at three levels: System on Chip, Electronic Control Unit, and system level. It emulates real systems, including hardware and software, efficiently managing increasing system complexity. A case study demonstrates the framework's ability to design, simulate, trace, profile, and debug AUTOSAR software using virtual platforms.

3 RESULTS

Integrating MiL, SiL, PiL, and HiL into the V-cycle provides a robust, iterative verification and validation process, enabling early error detection and continuous validation at various abstraction levels. The initial V-cycle is sequential and parallel, while the W-cycle features overlapping V-cycles, with verification cycles on both sides. Each development stage in the sequential line has a software verification cycle, maintaining continuous validation and verification. Sequentially, the initial V-cycle should be overlaid by the verification V-cycle with the following loops for each development stage:

- High-Level Design is initially addressed with MiL, performing initial simulations to verify core concepts and functionalities.
- Low-Level Design, MiL is also used, but with a focus on detailed model simulations, allowing the validation of specific system components.
- Software Development stage, Unit Testing is carried out using SiL, where the code is thoroughly verified.
- Software Integration, the Validation of Software Integration is conducted using SiL, ensuring that the different software modules work correctly together.
- Partial System Integration phase, PiL is employed to test the production code directly on the ECU hardware, allowing the identification of integration issues between software and hardware.
- System Verification stage, HiL is used for realistic validation and verification of the complete system with a simulated vehicle, replicating real operational conditions.

The W-cycle uses simulation loops for continuous and iterative verification at each development stage. V-INC involves iterative verification and validation with closed-loop testing during software development, aiming for cycles that resemble system verification. Virtual HiL combines HiL's advantages with virtual technology, enabling cost-effective use during development phases. This approach ensures continuous and efficient verification, rigorously testing automotive software from early development to full integration, significantly reducing risks and costs of late-detected failures.

4 FUTURE RESEARCH

Future research should implement closed-loop simulations in specific automotive software partitions, comparing software quality using metrics like early error detection, system robustness, correction efficiency, and cost-effectiveness. Additionally, exploring virtual HiL for early-stage verification could improve error detection, scalability, and economic benefits. This research could advance verification and validation practices in automotive software development.

ACKNOWLEDGMENTS

This work was partially supported by a grant from the National Council for Scientific and Technological Development (Grant CNPq-Universal 408651/2023-7).

REFERENCES

[1] S. Magnus Agren, Eric Knauss, Paolo Giusto, Grant Soremekun, Rogardt Heldal, and Daniela Damian. 2020. The Automotive Virtual Verification Ecosystem: Impediments and Enablers. *IEEE Software* 37, 5 (2020), 67–76. https://doi.org/10.1109/MS.2019.2905228

[2] Fabio Falcini and Giuseppe Lami. 2021. System and Software Testing in Automotive: an Empirical Study on Process Improvement Areas. In *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*. 253–262. https://doi.org/10.1109/ICST49551.2021.00035

[3] Sooyong Jeong, Yongsub Kwak, and Woo Jin Lee. 2016. Software-in-the-Loop simulation for early-stage testing of AUTOSAR software component. In *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*. 59–63. https://doi.org/10.1109/ICUFN.2016.7536980

[4] James Kapinski, Jyotirmoy Deshmukh, Xiaoqing Jin, Hisahiro Ito, and Ken Butts. 2015. Simulation-guided approaches for verification of automotive powertrain control systems. In *2015 American Control Conference (ACC)*. 4086–4095. https://doi.org/10.1109/ACC.2015.7171968

[5] Bohan Liu, He Zhang, and Saichun Zhu. 2016. An Incremental V-Model Process for Automotive Development. In *2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*. 225–232. https://doi.org/10.1109/APSEC.2016.040

[6] Florian Muttenthaler, Stefan Wilker, and Thilo Sauter. 2021. Lean automated hardware/software integration test strategy for embedded systems. In *2021 22nd IEEE International Conference on Industrial Technology (ICIT)*, Vol. 1. 783–788. https://doi.org/10.1109/ICIT46573.2021.9453538

[7] Mona Safar, Magdy A. El-Moursy, Mohamed Abdelsalam, Ayman Bakr, Keroles Khalil, and Ashraf Salem. 2019. Virtual Verification and Validation of Automotive System. In *Journal of Circuits, Systems and Computers*, Vol. 28. 1950071. https://doi.org/10.1142/S0218126619500713 arXiv:https://doi.org/10.1142/S0218126619500713

[8] Vassil Todorov, Frédéric Boulanger, and Safouan Taha. 2018. Formal Verification of Automotive Embedded Software. In *2018 IEEE/ACM 6th International FME Workshop on Formal Methods in Software Engineering (FormaliSE)*. 84–87.

[9] VDA QMC Working Group 13 / Automotive SIG 2017. *Automotive SPICE Process Assessment Model* (version 3.1 ed.). VDA QMC Working Group 13 / Automotive SIG. Available at: https://www.vda-qmc.de/en/publications/software-process-assessment-automotive-spice/.