Research on Automated Simulation Testing of Train Operation Monitoring and Recording Device

Xiaofan Mo[1,2], Yuhui He[1,2], Linfu Zhu[1,2,*], Jingyu Zheng[1,2] , Peng Jin[1,2]

[1]Standards & Metrology Research Institute, China Academy of Railway Sciences Corporation Limited, Beijing, 100081, China
[2]China Railway Inspection and Certification Center Co., Ltd, Beijing, 100081, China
*Corresponding author: zhulinfu2008@163.com

Abstract. The Train Operation Monitoring and Recording Device (LKJ) is a core onboard safety equipment for Chinese railways, and the correctness of its embedded control software directly affects train operation safety. Conventional manual testing suffers from low coverage, heavy workload, poor repeatability, and difficulty in defect reproduction. To solve these problems, the paper introduces a automatic simulation testing methodology for LKJ systems. The developed LKJ automated simulation testing system integrates three approaches of test case generation—graphical editing, manual operation recording, and LKJ operation record file conversion—to flexibly cover diverse scenarios. Key enabling technologies are analyzed systematically, including multi-type physical signal generation (speed, signal, pressure, etc.), DMI (Driver Machine Interface) input simulation via a robotic arm, and real-time acquisition of LKJ braking outputs and DMI display information based on video capture and optical character recognition. A timing delay calibration model is established to synchronize external signal injection with LKJ internal recording, improving testing accuracy. The system has been successfully applied in the development and verification of LKJ onboard software. The repeatability of test results reaches 100% under identical conditions. A practical technical method for automatic testing of LKJ and other railway train control systems is proposed, which increasesthe safety and reliability in railway operations.

Keywords: LKJ; Automated Simulation Testing; Physical Signal Generation; Signal Acquisition; Test Case Generation; HIL Test

1 Introduction

The Train Operation Monitoring and Recording Device (LKJ) is a fundamental onboard safety system and multiple-unit trains across China's railway network. It constantly monitors train speed, signal aspects, and driver actions, and automatically applies emergency braking when safety criteria are violated [1]. With its excellent performance and reliability, LKJ has become an indispensable technological pillar in the field of railway transportation. Figure 1 illustrates different application scenarios of LKJ. It plays a key role in freight transport—guaranteeing efficient and safe transport of cargo, and plays

L. W. Koon et al. (eds.), *Proceedings of the 2026 7th International Conference on Management Science and Engineering Management (ICMSEM 2026)*, Advances in Economics, Business and Management Research 399,
https://doi.org/10.2991/978-94-6239-758-3_7

a crucial role in passenger transport, supplying passengers with a safer and more comfortable trip. The LKJ system demonstrates its functions in shunting operations, where precise train speed control and operating status improves the reliability and safety.

Fig. 1. LKJ Application.

The LKJ onboard control software, which implements over 150 functional logic modules, is responsible for real-time decision-making; any software defect may lead to severe safety consequences. Therefore, comprehensive and effective test of LKJ software is a mandatory requirement throughout its lifecycle—from development and integration to application and maintenance [2].

Historically, LKJ software testing has been performed manually. Engineers design textual test cases, manipulate physical signal generators, and visually inspect DMI screens and recorded data files. While the method could verify basic functions, it is increasingly inadequate for LKJ, which contains complex logic, frequent updates, and shorter delivery cycles. Previous studies have reported that manual test typically achieves only 60–70% statement coverage and requires more than 30 person-hours per regression cycle [3], [4]. Moreover, human factors lead to inconsistent test execution and poor reproducibility of intermittent faults [5].

Researchers and manufacturers have proposed various automatic testing techniques for train control systems. He et al. developed a functional test platform for LKJ that partially automates signal injection and result collection, yet test case generation remains manual [6]. Bai et al. proposed a graphical test case definition method for LKJ, but the method could not integrate with real-time signal synchronization [7]. Outside the railway domain, model-based testing (MBT) and the test generation method based on search have been successfully applied to avionics and automotive embedded systems [8], [9]. However, direct adoption of these approaches to LKJ is hampered by the need for heterogeneous physical I/O, stringent real-time requirements and the intricacy of domain-specific signal logic. The deep learning algorithm has been adopted to generate GUI test scripts from the screenshots [10], but its application in the safety-critical human-machine interfaces (HMIs) of railways is still initial [11]. Hardware-in-the-loop (HIL) testing have been applied extensively in railway signaling test. [12], but the integration with LKJ software validation remains a challenging issue. The collaborative robotics for automatic test has exhibited high precision in industrial control panels [14], while advances in real-time signal synchronization further improve the accuracy of HIL simulations [15]. In addition, experimental studies on coverage-driven testing in safety-critical domains provide benchmarks that support our coverage improvement [16].

The paper presents a complete automatic simulation testing system specifically designed for LKJ. The main contributions are:

(1) The comprehensive architecture integrates test case generation, physical signal emulation, non-intrusive DMI operation, and multimodal output collection into a single workflow.

(2) Three test case acquisition methods covering both requirement-based and field-data-based scenarios, along with a timing-delay calibration model that guarantees accurate signal-injection timing.

(3) The quantitative evaluation method based on the extensive industrial application is proposed.

2 Test Process of LKJ

Testing of LKJ software follows two primary technical routes: manual test and automatic test. The both routes vary greatly in the technical features, implementation procedures.

2.1 Manual Test

The method includes drafting test cases and configuring dedicated test environments, manually configuring input physical signals, observing LKJ outputs and DMI (Driver Machine Interface) displays, and analyzing recorded operation data. Results are compared to determine the test conclusion [4].

The manual test limitations include repetitive data substitution resulting in heavy workload, textual test case descriptions causing interpretation variances among engineers, difficulty in reproducing discovered defects and difficulties in guaranteeing final test accuracy.

2.2 Automatic Test

Automatic testing addresses manual testing shortcomings by automatic test case execution, signal input through dedicated test systems [5]. Hardware-in-the-Loop (HIL) testing frameworks, which integrate real hardware with simulation models, are especially well-suited for complex cyber-physical systems like train controls, ensuring accurate signal synchronization and real-time performance [12].

2.2.1 Objectives of Automatic Testing.

Automatic testing supports three scenarios in the software lifecycle:

(1) Developer Debugging: Aims to validate the revised logic, the hybrid virtual-physical setups is employed in unit and integration to isolate defects [6].

(2) Independent Verification: Performed by testing departments to validate modifications of requirements and ensure non-regression of current functionalities conducted on physical hardware with parallel testing to improve efficiency.

(3) On-site User Acceptance: Targets validation under specific railway lines, employing focused testing on modified content and critical use cases.

2.2.2 General Process of Automatic Testing.

As illustrated in Figure 2, the automatic simulation testing process includes three stages:

(1) Test Case Generation: Creating executable test cases from requirements. AI and deep learning algorithm for automatic test, especially for GUI-based systems, represents that can enhance coverage [11].

(2) Test Execution: Interpreting test cases, driving signal generation equipment to produce physical inputs, controlling LKJ operation, and collecting all output data.

(3) Result Judgment: Automatically comparing LKJ outputs with anticipated results, flagging discrepancies for analysis.

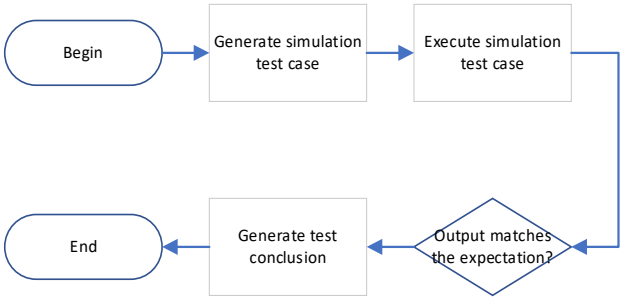

Fig. 2. General Process of LKJ Automated Simulation Testing.

2.3 Comparison with Existing Automatic Testing

To further highlight the originality, a systematic comparison between the proposed method and representative existing automatic test method for train operation and control system and related domains is conducted. Existing solutions either lack automatic test from field data, depend on intrusive bus simulation, or do not support comprehensive real-time synchronization.

3 Simulation Acquisition

Test cases form the foundation of automatic testing. Their scientific design and completeness directly determine testing effectiveness and precision [7]. Based on LKJ testing requirements, simulation test cases can be generated via four methods: scenario analysis, graphical editing, manual operation recording, and LKJ record file conversion.

3.1 Test Scenario

To gain an in-depth understanding of the complex application scenarios of LKJ—including high-speed railway, conventional speed railway, passenger transport, freight transport, and hub stations—a field investigation was conducted. The characteristics of each scenario are summarized as follows:

(1) High-Speed Railway Scenario

In high-speed railway scenarios, LKJ is required to achieve high-precision positioning and rapid response. Due to the high operating speeds, the real-time performance of LKJ is particularly stringent. Furthermore, the signaling systems in high-speed railways often adopt advanced train operation and control systems (such as ATP), and LKJ should closely cooperate with the system to ensure the reliable and controllable train operation at high speeds.

(2) Conventional Speed Railway Scenario

In conventional speed railway scenarios, LKJ primarily focus on the locomotive signal recognition, speed control, and monitoring of track conditions. The line conditions in conventional speed railways are diverse, including straight sections, curves, and gradients. LKJ should adjust its monitoring strategies according to the track conditions (including line changes, etc.) to ensure the smooth operation of the train.

(3) Passenger Transport and Freight Transport Scenarios

In passenger transport scenarios, in addition to ensuring train operation safety, LKJ also needs to take the passenger comfort experience into account. In the train acceleration, deceleration, and curve negotiation, LKJ should optimize its control strategies to reduce passenger discomfort. In freight transport scenarios, LKJ is required to monitor operation status and guarantee the transportation safety.

(4) Station Scenario

Station scenarios represent one of the most complex scenario in railways. In hub stations, LKJ must perform route control, signal interlocking, and collaborative operations with other systems. When the train enters or departs from stations, switch tracks, or passes through complex turnouts, LKJ must precisely control train movements to guarantee safe and efficient operation within the hub station.

3.2 Graphical Editing

The method abstracts testing functions (e.g., control condition operation, device operation, information acquisition, result judgment) into independent modules. Users genarates cases by drag-and-dropping logical controls, setting parameters, and defining control relationships.

Algorithm 1 shows the core process for generating a test case TC from graphical elements G:

Algorithm 1: Test Case Generation via Graphical Editing
Input: Set of graphical modules $G = \{g_1, g_2, \ldots, g_n\}$
Output: Executable case TC

1. **for each** module $g_i \in G$ **do**
2. Extract functional attributes $A(g_i)$
3. Map $A(g_i)$ to script command C_i
4. **end for**
5. Parse user-defined control relationships R between modules
6. **for each** relationship $r_{jk} \in R$ **do**
7. Apply timing between C_j and C_k
8. **end for**
9. Serialize ordered command set $\{C\}$ into TC
10. **return** TC

3.3 Manual Operation Recording

The method records a manual test process to generate a reusable automatic case. Steps include: recording the environment configuration, defining judgment items based on requirements, logging all interactions (clicks, settings, timestamps), capturing LKJ test results, and standardizing the recorded data into an executable case.

3.4 LKJ Operation Record File Conversion

LKJ devices record comprehensive operational data during real runs [8]. As shown in Figure 3, a record file consists of event rows containing name, time, mileage, and content. The conversion process includes analyzing the file, screening key events, and retrieving key parameters to form standardized test cases,which is shown in Table 1.

A critical challenge is the timing delay t_d, representing the lag from the signal generation to its recording. Different events have varying t_d (up to 5s). Direct use of raw timestamps can misalign signal input, affecting test accuracy. Therefore, a calibrated event execution timing strategy is essential, adjusting each event execution time in the case by $-t_d$.

The timing adjustment for an event E_i with original timestamp T_i^{record} can be modeled as:

$$T_i^{execute} = T_i^{record} - t_d(E_i) \tag{1}$$

where T_i^{record} is the timestamp in the log, and $t_d(E_i)$ is the empirically calibrated average delay for E_i. For each variant,a calibration process is performed by a reference signal to determine $t_d(E_i)$. The model guarantees that test replay accurately reconstructs the real-world timing sequence. Similar delay compensation techniques have been successfully adopted in other real-time embedded system testbeds [17].

I System A-mode Main Control Software	15:49:00	1454
I System B-mode Main Control Software	15:49:00	1454
II System A-mode Main Control Software	15:49:00	1454
II System B-mode Main Control Software	15:49:00	1454
I System B-mode Control Parameters	15:49:00	1454
I System B-mode Control Parameters	15:49:00	1454
II System A-mode Control Parameters	15:49:00	1454
II System B-mode Control Parameters	15:49:00	1454
I System A-mode Vehicle-mounted Data	15:49:00	1454

Fig. 3. Example of LKJ Operation Record File.

Table 1. Example of LKJ Automatic Test

Serial Number	Event Name	Occurrence Time	Mileage (km)	Event Parameters
1	Speed Change	12:29:30	12.345	Speed=5 km/h
2	Signal Change	12:29:35	12.458	Signal=Green Light
3	Speed Limit Change	12:30:50	15.678	Speed Limit=41 km/h

4 Execution of Simulation

The test execution is the essential element of automatic test, including four steps: external signal generation, device operation, output collection, and result verification.

4.1 Generation of External Physical Signals

LKJ operation requires multi-type signals, including the speed information, the locomotive signal, the handle position, the pipe pressure. These signals are got from the specialized equipment or the signal generators. The equipments produce FSK or specific waveforms signal.

The LKJ device requires several classes of analog and digital input signals.

(1) Speed signals: Frequency shift keying (FSK) pulses proportional to wheel rotation.

(2) Locomotive signals: AC/DC voltage levels indicating track circuit status.

(3) Handle position signals represent the driver's throttle and brake handle.

(4) Pipe pressure signals: 4–20 mA current loops for brake pipe pressure.

The test bench supports two types of signal generators.

(1) Specialized LKJ diagnostic instruments – native signal compatibility.

(2) Arbitrary function generators –A dedicated protocol adapter is proposed that translates high-level test commands into the appropriate signal generator commands.

4.2 Operation Control of LKJ Device

The operation control simulates the driver input, including the parameter entry, the notice input, the key presses, and the data dump procedures. The essensial is simulating DMI button press. Figure 4 illustrates the architecture of the LKJ automatic test system. Two technical schemes exist.

(1) Bus Communication: Interfacing with the DMI button board bus. The method allows both manual and software-triggered inputs but requires device modification and lacks cross-vendor compatibility.

(2) Robotic Arm: A 6-axis collaborative robot equipped with a soft silicone finger presses the DMI buttons exactly as a human. The approach is completely non-invasive, compatible with all LKJ variants, and is now the mainstream solution [13]. The robot is calibrated to the DMI screen coordinates using a vision-guided alignment procedure, achieving a positioning accuracy of ±0.5 mm[14].

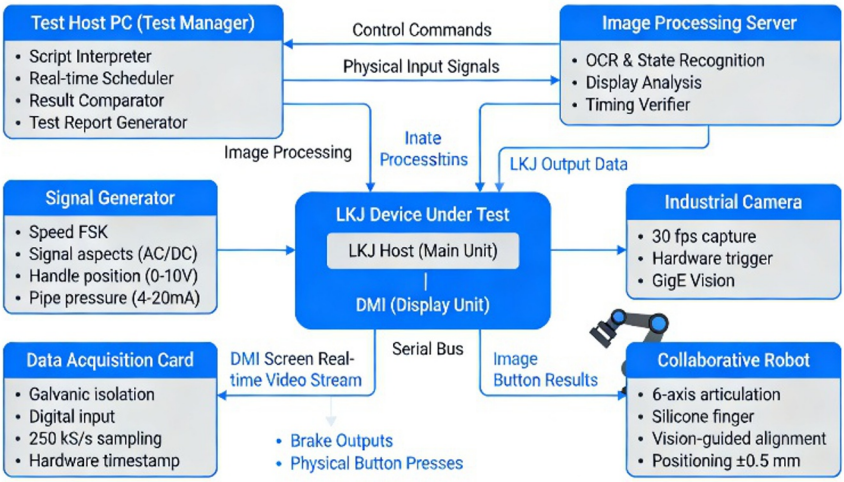

Fig. 4. Architecture of the LKJ automated simulation testing system.

4.3 Acquisition of LKJ Output Information

Output information focuses on the two information.

(1) Braking Output Signals: These signals are collected using professional signal acquisition modules connected to the corresponding interfaces of the LKJ device.

(2) DMI Display Information: Data are acquired through video capture and subsequent image analysis, with key indicators including signal type and number, real-time speed, speed limit, and braking curve.[13].

The video analysis process for extracting display text S from a DMI frame I can be summarized as:

(1) Region Detection: Locate dynamic information display area R_{info} in I.

(2) Preprocessing: Apply binarization and denoising to R_{info}.

(3) Character Segmentation & Recognition: Use OCR (Optical Character Recognition) to obtain raw text string S'.

(4) Semantic Parsing: Map S' to structured data S (e.g., {"speed": "80", "limit": "85"}).

Recent surveys confirm that such non-invasive HMI testing is gaining traction in safety-critical domains [11]. The novel deep learning-based OCR methods have demonstrated superior accuracy under lighting and reflection conditions [18].

4.4 Synchronization and Timing Control

Reliable hardware-in-the-loop simulation relies on deterministic timing.The LKJ automatic test system implements a multi-layer synchronization framework that coordinates signal generation, robotic actuation, and data acquisition with microsecond-level precision.

4.4.1 Clock Synchronization.

All test instruments support precision time protocol and are synchronized to the test host's grandmaster clock with an accuracy of ±1 μs. The network switches are PTP-transparent, and the boundary clocks are deployed to reduce cumulative drift. Every acquired sample and actuation event is tagged with a 64-bit global timestamp, enabling sample-accurate alignment across distributed components.

4.4.2 Timing Delay Characterization and Compensation.

LKJ record files exhibit an inherent input-to-log delay $t_d(E_i)$ that varies per event type. During system commissioning, a calibration procedure for each LKJ variant under test is performed.

(1) A reference pulse is injected via the signal generator at a known absolute time T_{inj}.

(2) The LKJ's internal event log is downloaded and the recorded timestamp T_{rec} of the corresponding event is extracted.

(3) The delay is computed as $t_d = T_{\text{rec}} - T_{\text{inj}}$.

(4) The measurement is repeated to obtain the mean and standard deviation.

The typical calibrated delays range from 0.8 s to 4.7 s. The standard deviation is below 150 ms for all event classes, indicating stable behavior. The derived t_d values are stored in a configuration database and applied to each converted test case via Equation (1). Similar timestamp calibration techniques are widely applied in real-time embedded system testbeds [17].

4.4.3 Runtime Triggering and Watchdog.

During test execution, the scheduler sends control commands to both the signal generator and robot controller over a deterministic Ethernet link. Each device replies with a hardware-timestamped response, which is then checked against the predefined schedule. In parallel, a software watchdog thread supervises the execution progress of each individual test step. (1) Signal outputs: If the system does not detect the expected braking relay state within ±100 ms of the programmed time, the test is paused and an alert is raised.

(2) DMI display changes: The vision system output is expected within ±1 s of the scripted event.

4.4.4 Multi-Bench Coordination.

The test setup consists of three identical units operating in parallel to speed up regression testing. Each unit is equipped with an independent local real-time controller, while a central orchestrator assigns test tasks and aggregates the experimental results. Phase locking between units is not necessary, yet all clocks are traceable to the same stratum-1 NTP server. The inter-bench jitter is maintained below 50 μs, making it negligible within the system's ±100 ms tolerance range. [15].

The timing performance of the test setup accurately reflects real-world conditions, enabling the robust detection of time-critical software defects.

5 System Implementation and Application

The proposed LKJ automatic simulation testing method has been fully implemented and deployed. It consists of three physical test benches (each equipped with an LKJ device, a robotic arm, signal generators, and acquisition modules) and a central test management server. This system has been applied to daily developer debugging. Test duration has been reduced from 32 hours in manual test to 7.7 hours using automatic testing, which is achieved across three test benches.

6 Conclusion

This paper presents an in-depth investigation into LKJ testing methods and core automatic simulation technologies, on the basis of which an LKJ-oriented automatic simulation test system is proposed. The successful application in LKJ software testing shows that the system alleviates the drawbacks of conventional manual testing, including low test coverage, heavy manual intervention, insufficient repeatability, and the difficulty in reproducing faults. A practical technical solution is proposed for the automatic testing of LKJ, which increasing the safety and reliability of train operation.

Acknowledgments

This work was supported by the Science and Technology Research and Development Program of China State Railway Group Co., Ltd. (Project Name: Research on Test Verification Technology Based on the Integration of Digital Simulation and Experimentation ; Grant No. K2024B003).

References

1. J. M. Zhang, M. Harman, L. Ma, and Y. Liu, "Machine Learning Testing: Survey, Landscapes and Horizons," IEEE Transactions on Software Engineering, vol. 48, no. 1, pp. 1-36, Jan. 2022, doi: 10.1109/TSE.2019.2962027.
2. H. Song, Y. Liu, S. Liu, and T. Tang, "Formal Modeling and Verification Methods for the System Requirement Specifications of Train Control Systems: A Survey," IEEE Transac-

tions on Intelligent Transportation Systems, vol. 26, no. 2, pp. 1419-1440, Feb. 2025, doi: 10.1109/TITS.2024.3513717.
3. L. Barruffo, B. Caiazzo, A. Petrillo, and S. Santini, "A GoA4 Control Architecture for the Autonomous Driving of High-Speed Trains Over ETCS: Design and Experimental Validation," IEEE Transactions on Intelligent Transportation Systems, vol. 25, no. 5, pp. 3576-3591, May 2024, doi: 10.1109/TITS.2023.3338295.
4. A. Paspatis, A. Kontou, Z. Feng, M. Syed, G. Lauss, G. Burt, P. Kotsampopoulos, and N. Hatziargyriou, "Virtual Shifting Impedance Method for Extended Range High-Fidelity PHIL Testing," IEEE Transactions on Industrial Electronics, vol. 71, no. 3, pp. 2903-2913, Mar. 2024, doi: 10.1109/TIE.2023.3269467.
5. A. Memon, I. Banerjee, and A. Nagarajan, “What test oracle should I use for effective GUI testing?,” in Proc. IEEE 18th Int. Conf. Automated Softw. Eng., 2003, pp. 164–173, doi: 10.1109/ASE.2003.1240304
6. M. Utting, A. Pretschner, and B. Legeard, “A taxonomy of model-based testing approaches,” Software Testing, Verification and Reliability, vol. 22, no. 5, pp. 297–312, 2012, doi: 10.1002/stvr.456.
7. J. M. Zhang, M. Harman, L. Ma, and Y. Liu, "Machine Learning Testing: Survey, Landscapes and Horizons," IEEE Transactions on Software Engineering, vol. 48, no. 1, pp. 1-36, Jan. 2022, doi: 10.1109/TSE.2019.2962027.
8. A. Bertolino, B. Miranda, R. Pietrantuono, and S. Russo, "Adaptive Test Case Allocation, Selection and Generation Using Coverage Spectrum and Operational Profile," IEEE Transactions on Software Engineering, vol. 47, no. 5, pp. 881-898, May 2021, doi: 10.1109/TSE.2019.2906187.
9. H. Ren, H. Li, Z. Wang, and R. Lu, "Cloud-Based Distributed Group Asynchronous Consensus for Switched Nonlinear Cyber-Physical Systems," IEEE Transactions on Industrial Informatics, vol. 21, no. 1, pp. 693-702, Jan. 2025, doi: 10.1109/TII.2024.3457811.
10. P. K. Andyartha, B. D. Mardiana, U. Hasan, N. Elqolby and D. Siahaan, "Improving Mobile Application GUI Testability with Deep Learning-based Test Case Generation," 2023 International Workshop on Artificial Intelligence and Image Processing (IWAIIP), Yogyakarta, Indonesia, 2023, pp. 28-33, doi: 10.1109/IWAIIP58158.2023.10462806.
11. S. Kandl, R. Kirner and P. Puschner, "Development of a Framework for Automated Systematic Testing of Safety-Critical Embedded Systems," 2006 International Workshop on Intelligent Solutions in Embedded Systems, Vienna, Austria, 2006, pp. 1-13, doi: 10.1109/WISES.2006.329116.
12. J. del Olmo, J. Poza, F. Garramiola, T. Nieva and L. Aldasoro, "Model driven Hardware-in-the-Loop Fault analysis of railway traction systems," 2017 IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM), Donostia, Spain, 2017, pp. 1-6, doi: 10.1109/ECMSM.2017.7945878.
13. H. V. P. Singh and Q. H. Mahmoud, "HMI-guard: A platform for detecting errors in human-machine interfaces," 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Banff, AB, Canada, 2017, pp. 2861-2866, doi: 10.1109/SMC.2017.8123061.
14. Tufano, F ; Bahadure, SW; Tufo, M ; Novella, L ; Fiengo, G and Santini, S, "An Optimization Framework for Information Management in Adaptive Automotive Human-Machine Interfaces," APPLIED SCIENCES-BASEL,vol.13,no.19,p.10687, 2023, doi: 10.3390/app131910687.
15. E. Guillo-Sansano, M. H. Syed, A. J. Roscoe, G. M. Burt and F. Coffele, "Characterization of Time Delay in Power Hardware in the Loop Setups," in IEEE Transactions on Industrial Electronics, vol. 68, no. 3, pp. 2703-2713, March 2021, doi: 10.1109/TIE.2020.2972454.

16. Islam, G; Storer, T , “A case study of agile software development for safety-Critical systems projects,” RELIABILITY ENGINEERING & SYSTEM SAFETY, vol.200, 2020, doi:10.1016/j.ress.2020.106954.
17. Y. Li, M. Liu, H. Ren, A. Mishchenko, and C. Yu, "DAG-Aware Synthesis Orchestration," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 43, no. 12, pp. 4666–4675, Dec. 2024, doi: 10.1109/TCAD.2024.3397052.
18. Q. Song, Y. Liu, H. Sun, Y. Chen, and Z. Zhou, "Multi-Device Universal Automation Data Acquisition and Integration System," IEEE Access, vol. 12, pp. 106705–106716, 2024, doi: 10.1109/ACCESS.2024.3435386.