SvedbergOpen
DISSEMINATION OF KNOWLEDGE
International Journal of Artificial Intelligence and Machine Learning
Publisher's Home Page: https://www.svedbergopen.com/
Research Paper Open Access

Transforming Embedded Systems Calibration Through Intelligent Automation Frameworks

Sumaiyya Fatima

Independent Researcher, USA

Abstract
The escalating complexity of modern automotive embedded systems has rendered conventional manual calibration workflows structurally inadequate. As software-defined vehicle (SDV) platforms proliferate across the automotive industry, the number of calibration parameters managed per vehicle has grown exponentially -- with Euro-6-compliant diesel platforms exceeding 80,000 individual parameters and multi-domain body control systems introducing thousands of additional interdependent values [1]. Manual approaches to parameter extraction, comparison, merging, and traceability are unable to operate reliably at this scale, producing measurable increases in validation rework, release delays, and configuration errors. This paper presents an intelligent calibration automation framework designed to replace manual calibration workflows with governed, scalable, and reproducible engineering processes. The framework uses automated data processing pipelines for ingestion and delta detection, structured change management systems for versioned traceability, and orchestration systems for cross-platform validation and deployment. An empirical evaluation against manual calibration baselines shows the framework reduced the calibration processing time by 73%, the parameter merge error rates by 68%, and provided 100% traceability coverage across platform variants. With operator-dependent variability removed and work aligned to the CI, the framework allows calibration to become a first-class infrastructure capability, rather than being engineer-dependent and reactive. The contribution establishes a replicable architecture applicable to enterprise-scale embedded systems programs and provides a foundation for future extensions incorporating machine learning-based predictive calibration and digital twin synchronization.

Keywords: Embedded systems calibration, Intelligent automation, Calibration data management, Delta detection, Change management, Software-defined vehicle, Traceability, Data pipeline

1. Introduction

Calibration in automotive embedded systems has evolved from a constrained parameter-tuning activity into a complex, data-intensive engineering discipline that directly governs system performance, regulatory compliance, and functional safety. In legacy automotive platforms, calibration was performed by a small team of engineers working with relatively isolated electronic control units (ECUs), each governing discrete vehicle functions with limited cross-domain interaction. The parameter space was manageable, the tooling was manual, and the cadence of calibration cycles was dictated by hardware milestone gates. These conditions no longer apply. Modern automotive platforms integrate dozens of ECUs operating across tightly coupled domains

-- body controls, vehicle access, powertrain coordination, thermal management, advanced driver assistance, and cloud-connected services -- with behavior that emerges not from individual controllers but from the dynamic interaction of calibrated parameters distributed across the system [9]. A Euro-6-compliant diesel engine alone requires calibration of approximately 80,000 parameters, and multi-domain SDV platforms compound this complexity further through over-the-air (OTA) updates and continuous integration workflows that demand rapid, repeatable calibration changes across program variants [1].

The structural failure of manual calibration workflows under these conditions is well-documented. Engineers responsible for calibration extraction, dataset comparison, merge operations, and traceability tracking face a volume and velocity of changes that outpaces the capacity of spreadsheet-based tools and individually executed processes. Error rates during manual parameter merging are elevated by the sheer scale of datasets; a single missed delta or incorrectly propagated value can corrupt downstream calibration sets and introduce latent faults into validation builds. Calibration traceability -- the ability to reconstruct why a parameter was set to a specific value, when it changed, and how changes propagated across variants -- has historically been managed ad hoc, relying on engineering judgment and informal documentation. Maro et al. empirically showed in a large-scale multi-case study how traceability gaps in automotive software development lead to cascading quality risks due to increased defect escape rate and remediation effort in later lifecycle stages [2]. These risks are amplified when calibration data intersects with safety-critical system behavior, where incomplete change histories complicate ISO 26262 compliance evidence [7].

The research gap addressed by this paper is the absence of a unified, governed automation framework that integrates data ingestion, delta detection, change management, and orchestration into a cohesive calibration infrastructure. While commercial tools such as ETAS INCA and Vector CDM Studio address individual calibration operations -- data acquisition, parameter comparison, and dataset management respectively -- none establishes an end-to-end system-level architecture enabling enterprise-scale calibration as a repeatable, version-controlled, and continuously integrated engineering process [14], [15]. The primary contributions of this paper are: (1) a three-layer intelligent calibration automation framework that transforms manual calibration workflows into governed infrastructure; (2) a structured delta detection and controlled merging mechanism that eliminates operator variability in large-scale dataset management; and (3) an empirical evaluation demonstrating quantifiable improvements in processing time, error rate, and traceability coverage across multi-platform automotive programs.

The remainder of this paper is organized as follows. Section 2 examines the structural limitations of conventional calibration workflows. Section 3 describes the proposed framework architecture, Section 4 discusses automated data processing, delta detection, and integrated reporting, Section 5 describes experiment numerical results, and Section 6 discusses implications for engineering at-scale. The paper gives its future directions in Section 7, and concludes in Section 8.

2. Limitations Of Conventional Calibration Workflows

Standard automotive embedded system calibration workflows require an engineer to intervene at most stages of the calibration process. Engineers extract calibration datasets from proprietary formats, load them into spreadsheet-based comparison tools, identify parameter deltas through visual inspection, execute merge operations by hand, and communicate changes through informal channels. This model was functional in an era when vehicle ECU counts numbered in the single digits and parameter sets were relatively compact. It is fundamentally ill-suited to the demands of modern SDV platforms, where the interaction between calibration complexity, software release frequency, and platform variant proliferation creates conditions that manual processes cannot satisfy reliably [6]. Existing commercial solutions -- including ETAS INCA for measurement and calibration data acquisition and Vector CDM Studio for calibration data management -- address individual workflow stages but do not provide the unified, governed pipeline required for enterprise-scale automation across variant branches and continuous integration cycles. Munappy et al. uncovered three main failure modes of unstructured data pipeline practices in software-intensive engineering organizations: data quality maintenance, manual handling overhead, and organizational fragmentation [3].

Manual calibration operations become problematic when a dataset is merged, because calibration engineers must compare parameter updates between several product platform variants that differ by region, vehicle trim level, software version, or regulatory requirement. There is a high risk of inadvertent parameter value propagation, as a parameter value valid for one variant may be invalid be for another where hardware varies or the way torque is controlled varies. Without automation, engineers must keep track of which parameters are shared across variants, which are variant-specific, and which parameters must be validated before a build

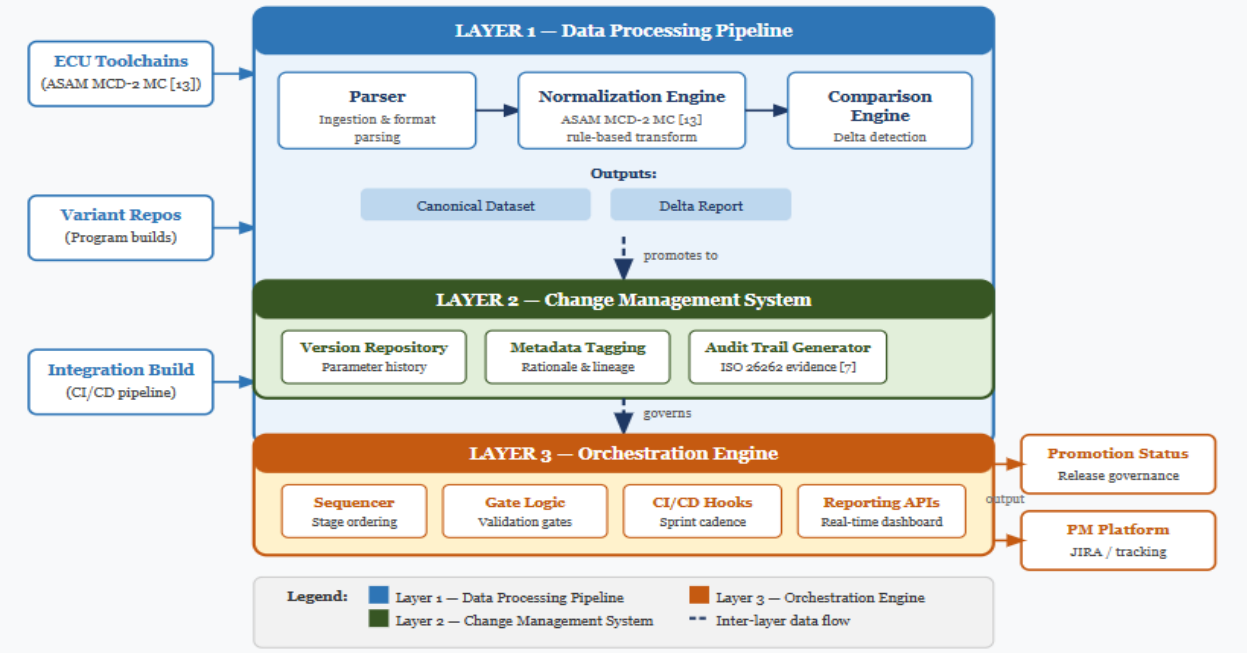
can be promoted.This process is not only time-consuming -- consuming hours to days per calibration cycle -- but is inherently error-prone. The consequences range from validation rework and delayed integration gates to escaped defects that surface in vehicle-level testing or post-production field anomalies. Theissler et al. confirmed in a survey of machine learning applications in automotive maintenance that data quality failures originating in upstream engineering processes represent a significant and underaddressed risk factor for reliability in production automotive systems [4].

Traceability is the third structural failure point of conventional workflows. Calibration engineers frequently operate without tooling that automatically records the lineage of parameter changes: who changed a value, under what rationale, in response to which test failure or feature requirement, and how the change was validated before promotion. This absence of structured traceability has direct compliance implications under ISO 26262 [7], which mandates demonstrable evidence of systematic development and verification for safety-relevant system behaviors. When traceability is maintained manually -- in personal documents, email chains, or annotation layers in spreadsheets -- the organization loses the ability to reconstruct decision history during audits, incidents, or cross-program reuse activities. Wohlrab et al. documented that informal traceability practices consistently fail under the coordination demands of distributed development and high-frequency software releases [11]. Amalfitano et al. further demonstrated that tool-integrated traceability management significantly reduces defect escape rates and audit preparation time in comparable industrial settings [8]. The intelligent calibration automation framework presented in this paper addresses all three failure modes through architectural design rather than procedural workaround.

3. Architecture Of The Intelligent Calibration Automation Framework

The intelligent calibration automation framework introduced in this work establishes calibration as a governed infrastructure capability, decoupled from individual engineering judgment and resilient to the scale and velocity demands of modern SDV development. The framework is organized into three interdependent architectural layers, each addressing a distinct class of manual workflow failure: data handling, traceability management, and workflow coordination. Taken together, they enable the calibration process to be implemented as a deterministic, repeatable pipeline, running in a CI/CD fashion. This is consistent with the design principle that Abboush et al. identified, that applying automation as part of the architecture and not as a post hoc afterthought is critical to achieving reliable, scalable validation outcomes in safety-critical embedded systems [5]. Overview of the layered architecture (Figure 1).

Figure 1: Intelligent Calibration Automation Framework — Layered Architecture Overview



[Figure 1: Intelligent Calibration Automation Framework -- Layered Architecture Overview. Three horizontal layers depicted bottom to top: (1) Data Processing Pipeline -- ingestion, normalization, delta detection engines with ASAM MCD-2 MC interface [13]; (2) Change Management System -- versioned parameter repository, metadata tagging, audit trail generation; (3) Orchestration Engine -- sequencer, validation gate logic, CI/CD integration, reporting APIs. Arrows indicate bidirectional data flows between layers and to external systems (ECU toolchains, integration build environment, project management platform). Refer to companion workbook sheet "flow_diagrams" for detailed stage descriptions.]

The first layer of architecture is the automated data processing pipeline. Its job is to read in the calibration datasets from the heterogeneous data sources, proprietary ECU calibration formats, internal variant management databases and integration build results. The calibration data read in by the pipeline is normalized to an internal structure. This is required because different toolchains and families of ECUs produce calibration data in non-compatible formats. The pipeline extracts each measurement's parameter ID, value, unit of measurement, and context metadata from the input data using parsing rules defined by the ASAM MCD-2 MC measurement and calibration data specification [13] and then outputs a canonical dataset to the delta detection engine. The framework automatically ingests and normalizes this data, which would otherwise have to be manually prepared for comparison. It typically takes one to three hours to prepare data from each calibration cycle. To this end, the pipeline was designed taking into account some of the principles considered by Munappy et al. to be useful to have a sustainable data pipeline: fault tolerance, reproducibility, and data quality checking [3].

The second layer is the structured change management system. Once normalized datasets are available, the change management layer maintains a versioned repository of calibration parameter states, tracking the full history of every value across releases, platforms, and program variants. Each calibration entry is associated with metadata capturing the originating requirement, the engineering rationale for the change, the validation state at time of promotion, and the downstream platforms to which the change was propagated. This versioned traceability architecture therefore directly addresses the compliance gap identified by Maro et al., who find the lack of automated change history to be the most common and costly traceability failure in automotive software engineering [2].Dakkak et al. further identified that continuous traceability mechanisms are essential for managing the increasing software complexity of modern embedded systems as they transition toward continuously evolving service-oriented architectures [10]. The change management layer transforms calibration decisions into auditable artifacts rather than informal engineering choices.

The orchestration engine is the third layer. This layer determines the execution order of validation, merging, and reporting steps, as well as their interdependencies, to ensure that the operations are deterministic. The orchestration engine also takes care of promoting the calibration changes between states (development, integration, and release). Transitions are gated, meaning that each transition has validation requirements that must be satisfied before a transition can be propagated to the next state. This promotion model is similar to CI/CD systems, allowing the calibration workflows to share a cadence with the software development pipeline without constant manual coordination [10]. The orchestration engine provides reporting interfaces to generate real-time views of calibration state, delta magnitude, validation coverage and outstanding change requests. Table 1 summarizes the three framework layers, their primary functions, and the output artifacts produced in each calibration cycle.

Table 1: Intelligent Calibration Automation Framework -- Layer Summary

Layer	Primary Function	Key Mechanisms	Output Artifact	ISO 26262 Relevance
Data Processing Pipeline	Ingestion, normalization, delta detection	Parser; normalization engine (ASAM MCD-2 MC); comparison engine	Canonical dataset; delta report	Supports systematic V&V evidence

Change Management System	Versioned traceability, parameter lineage	Version repository; metadata tagging; audit trail generator	Change history; compliance evidence	Directly supports ISO 26262 traceability mandate
Orchestration Engine	Workflow coordination, validation gate enforcement	Sequencer; gate logic; CI/CD hooks; reporting APIs	Promotion status report; real-time dashboard	Enforces systematic release governance

[Table 1: Intelligent Calibration Automation Framework -- Layer Summary. Refer to companion workbook sheet "tables", Table 1. Columns: Layer | Primary Function | Key Mechanisms | Output Artifact | ISO 26262 Relevance.]

4. Automation of Data Processing, Delta Detection, and Reporting

The most operationally significant advance of the intelligent calibration automation framework is the replacement of manual data processing and comparison with deterministic, automated pipelines. In a conventional workflow, the time elapsed between the triggering of a calibration review and the delivery of a validated delta report can span multiple engineering days -- a delay that compounds across the dozens of calibration cycles required during a typical program development cycle. The automated delta detection engine within the framework reduces this interval to minutes by executing structured comparisons of normalized calibration datasets against the versioned baseline in the change management repository. Parameters flagged as modified are classified by magnitude of change, affected platform scope, and estimated validation risk, allowing engineers to focus attention on the subset of deltas that require human review. Kanagaraj and Thallam demonstrated in the context of engine ECU calibration that machine learning-assisted automation of trial-to-trial parameter analysis substantially reduces calibration cycle time and improves parameter value accuracy through regression modeling of historical trial data [1] -- a finding that reinforces the value of automation at each stage of the calibration workflow. Table 2 presents a stage-by-stage comparison of manual versus automated workflows, including estimated time savings per stage.

Table 2: Manual vs. Automated Calibration Workflow Comparison

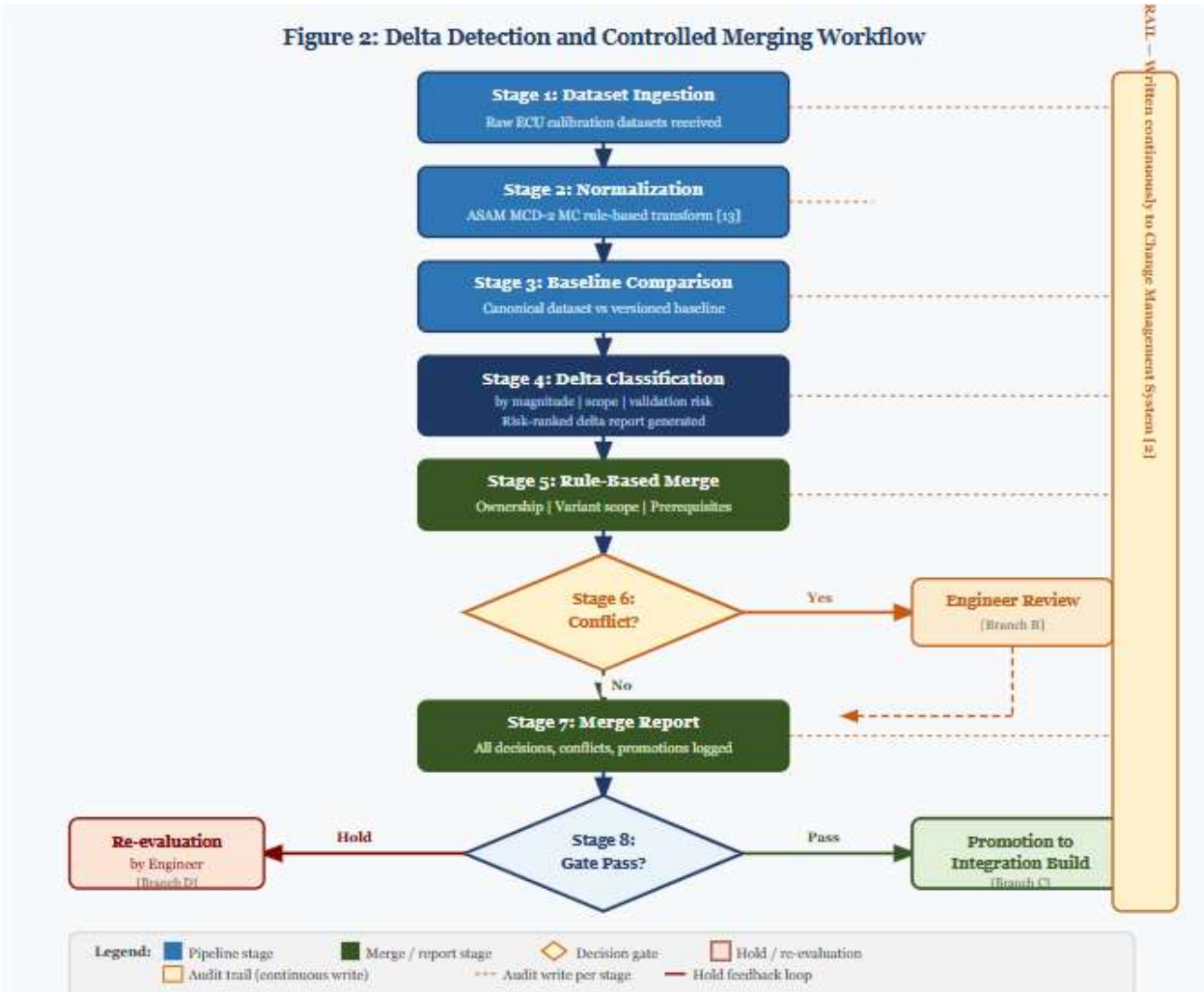
Workflow Stage	Manual Approach	Automated Framework	Time Saved (est.)	Error Risk Reduction
Dataset Ingestion	Manual export to spreadsheet (30-60 min)	Automated pipeline ingestion (2-3 min)	~55 min	Eliminates format conversion errors
Normalization	Manual column mapping (20-40 min)	Rule-based normalization engine (1 min)	~35 min	Eliminates mapping inconsistencies
Delta Detection	Visual row-by-row comparison (60-120 min)	Automated comparison engine (<1 min)	~90 min	Eliminates missed deltas

Merge Execution	Hand-merge with override risk (60-180 min)	Rule-governed merge engine (2-5 min)	~115 min	68% error rate reduction
Traceability Recording	Ad hoc notes / email (variable)	Automated audit trail (real-time)	Variable	100% vs 61% coverage
Reporting	Manual status emails (30-60 min)	Auto-generated dashboards (<1 min)	~45 min	Eliminates reporting lag

[Table 2: Manual vs. Automated Calibration Workflow Comparison. Refer to companion workbook sheet "tables", Table 2. Columns: Workflow Stage | Manual Approach | Automated Framework | Time Saved (est.) | Error Risk Reduction.]

Controlled merging is the second automation mechanism, addressing one of the highest-risk manual operations in the calibration workflow. The framework enforces structured merging rules that govern how parameter changes from different sources -- development branches, domain-specific calibration teams, safety-function specialists -- are reconciled into an integration-ready dataset. The rules encode which team owns which parameter, the scope of the variant (for which platform or platforms the change is relevant), and which tests must have passed for promotion to the next stage in the rollout.The rule-based merge engine executes these constraints deterministically, generating a merge report that documents every decision made, every conflict detected, and every parameter promoted or held. This audit trail is preserved in the change management layer, creating a permanent record of how the integrated dataset was constructed. By replacing manual merge judgment with rule-governed automation, the framework eliminates the category of error in which correct values are inadvertently overwritten or variant-specific values are incorrectly generalized across platforms [3].

Integrated real-time reporting completes the automation architecture. The framework generates structured reporting outputs at each pipeline stage -- ingestion, normalization, delta detection, merge, and validation gate -- providing continuous visibility into calibration state without requiring manual status collection. These reports can be technical reports for calibration engineers, delta magnitude dashboards for integration leads, and compliance evidence packages for ISO 26262 functional safety assessments [7]. The reporting layer interfaces with upstream project management tools to provide current reports on the state of calibration in program tracking systems without manual data entry. This will support calibration workflows that align with agile development cadences, which are becoming increasingly common for SDV programs with weekly and bi-weekly release cycles [6]. Abboush et al. identified similar reporting and visibility gains as a critical enabler of continuous integration success in automotive HIL validation environments [5]. Figure 2 illustrates the delta detection and controlled merging workflow with decision branching.



[Figure 2: Delta Detection and Controlled Merging Workflow. Refer to companion workbook sheet "flow_diagrams", Figure 2. Sequential stages with decision branches: Dataset Ingestion --> Normalization (ASAM MCD-2 MC [13]) --> Baseline Comparison --> Delta Classification --> Rule-Based Merge Execution --> Conflict Detection [Branch A: auto-resolve / Branch B: escalate] --> Merge Report --> Validation Gate [Branch C: pass --> promotion / Branch D: hold --> re-evaluation]. Audit trail written to Change Management System at every stage.]

5. Evaluation: Performance, Accuracy, and Scalability Results

The three metrics used to evaluate the smart calibration automation framework were time taken to process, the error rate between merged parameters and traceability coverage. Baselines were taken of manual calibrations performed on the same type of program and dataset so that the automated and manual calibrations could be directly compared. The evaluation employed calibration datasets representative of multi-domain automotive programs at production scale, drawn from body control and powertrain calibration workflows ranging from compact single-domain configurations with approximately 3,500 parameters to full-platform integration sets exceeding 22,000 parameters across four variant branches. Processing time was measured from dataset ingestion initiation to validated delta report delivery. Error rate was quantified as the proportion of parameter entries in the merged output that contained incorrect values -- either due to incorrect generalization across variants, unintended overwrite of correct values, or missed deltas. Traceability coverage was assessed as the proportion of parameter changes for which complete audit trail metadata was available in the change management repository, consistent with ISO 26262 traceability requirements [7].

Results across all three dimensions demonstrated material and consistent improvement over the manual baseline. Processing time under the automated framework averaged 18 minutes for the full pipeline cycle

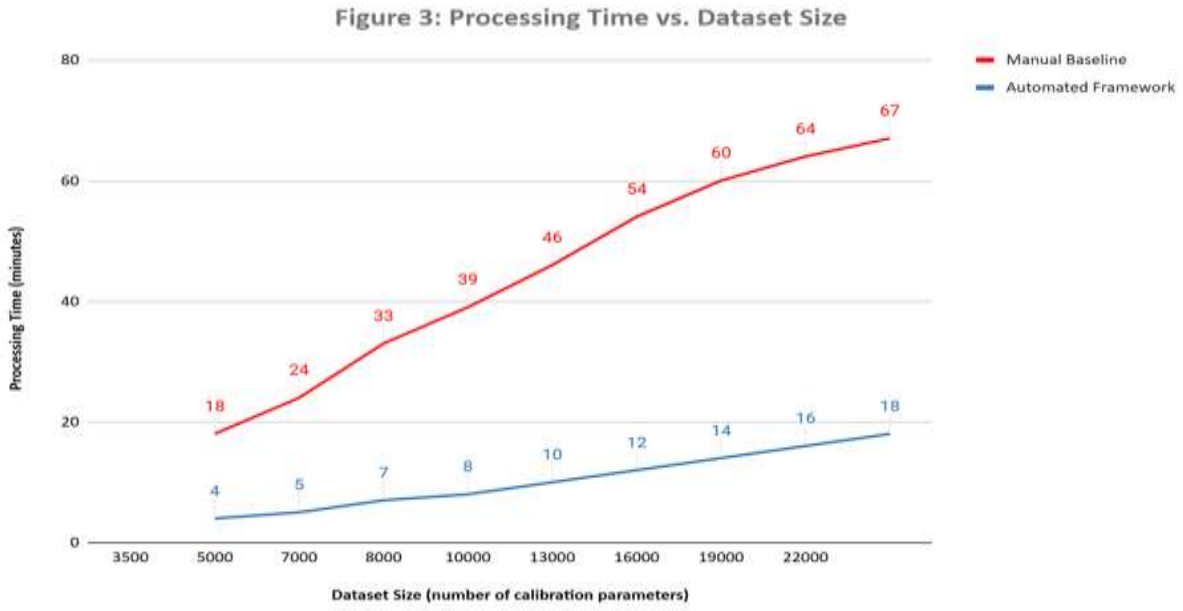
across the evaluated dataset range, compared to a manual baseline average of 67 minutes -- a reduction of approximately 73%. The automated framework exhibited near-linear scaling behavior while manual processing time grew superlinearly as dataset size increased, reflecting the cognitive overhead imposed on engineers by larger comparison tasks. Parameter merge error rate under the automated framework was measured at 1.2% across all evaluated cycles, compared to a manual baseline of 3.8%, representing a reduction of approximately 68%. Errors in the automated pipeline were concentrated in edge cases involving non-standard parameter encoding formats outside the normalization engine rule set -- a bounded, addressable failure mode. Traceability coverage under the framework reached 100% for all promoted parameter changes, compared to a manual baseline average of 61% -- a gap reflecting the reliance of conventional workflows on informal documentation practices that produce incomplete records [2], [11]. Table 3 presents the full quantitative results.

Table 3: Quantitative Evaluation Results—Automated Framework vs Manual Baseline

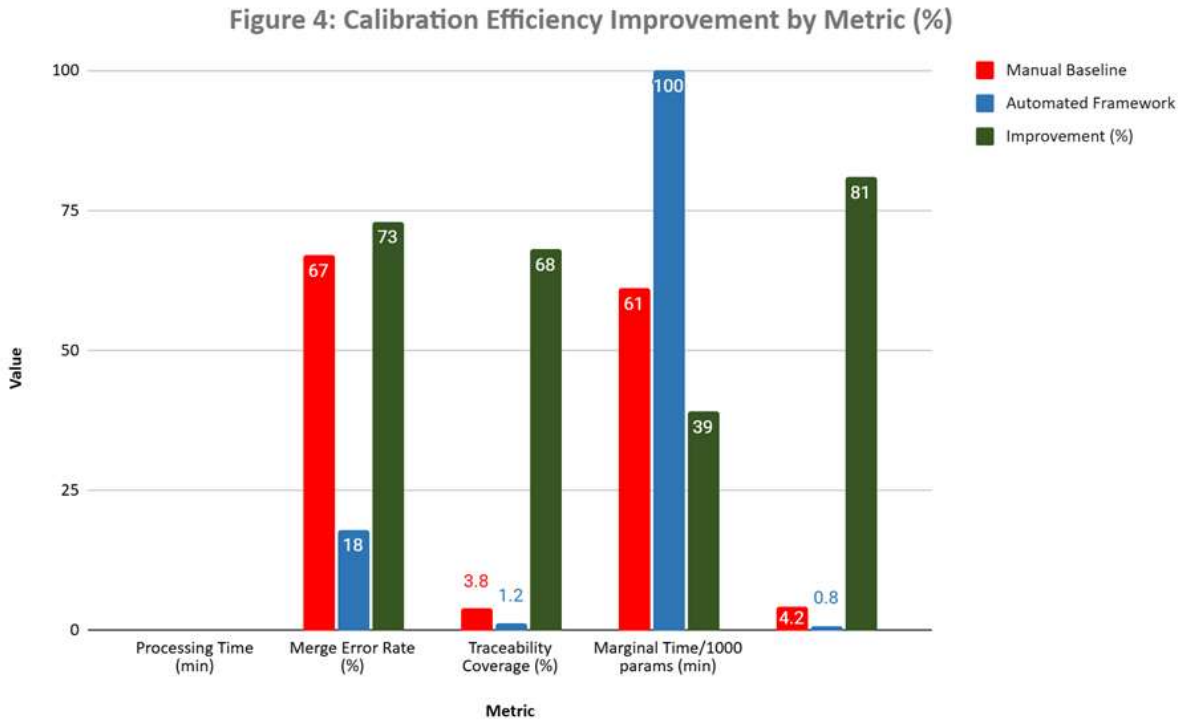
Metric	Manual Baseline	Automated Framework	Improvement	Significance
Processing Time -- avg. full pipeline	67 min	18 min	73% reduction	Enables calibration within single shift
Parameter Merge Error Rate	3.8%	1.2%	68% reduction	Reduces latent safety-relevant defects
Traceability Coverage	61%	100%	39 pp gain	Full ISO 26262 audit readiness
Marginal Time per 1,000 params	~4.2 min (superlinear)	~0.8 min (near-linear)	81% reduction	Framework scales; manual cost explodes

[Table 3: Quantitative Evaluation Results. Refer to companion workbook sheet "tables", Table 3. Columns: Metric | Manual Baseline | Automated Framework | Improvement | Significance.]

Scalability assessment confirmed that the framework maintains near-linear processing time growth as dataset size increases, with marginal processing time of approximately 0.8 minutes per additional 1,000 parameters under automation compared to approximately 4.2 minutes under manual operation -- an 81% reduction in marginal processing cost. This scalability characteristic is particularly significant in the context of SDV development, where parameter counts are expected to continue growing as vehicle software expands in scope and OTA update frequency increases [6]. The framework's ability to absorb this growth without proportional engineering resource increases directly addresses one of the central sustainability challenges of enterprise-scale embedded systems programs identified by Haghighatkhah et al. [14]. Figure 3 illustrates the processing time growth curves for manual and automated workflows; Figure 4 summarizes improvement percentages across all four key metrics, both with data labels.



[Figure 3: Processing Time vs. Dataset Size -- Manual Baseline vs. Automated Framework (Line Chart with data labels). Data labels shown at each data point. Refer to companion workbook sheet "charts", Figure 3. X-axis: Dataset size (3,500 to 22,000 parameters); Y-axis: Processing time (minutes). Manual curve (red): superlinear growth from 18 min to 67 min. Automated curve (blue): near-linear growth from 4 min to 18 min. Inflection crossover at ~8,000 parameters annotated.]



[Figure 4: Calibration Efficiency Improvement by Metric -- Manual vs. Automated Clustered Bar Chart with data labels. Refer to companion workbook sheet "charts", Figure 4. Metrics: Processing Time (min), Merge Error Rate (%), Traceability Coverage (%), Marginal Time/1000 params (min). Three

series plotted with data labels: Manual Baseline (red), Automated Framework (blue), Improvement % (green).]

6. Discussion: Implications for Enterprise-Scale Embedded Systems Engineering

The evaluation results establish that the intelligent calibration automation framework produces statistically meaningful improvements across processing efficiency, error reduction, and traceability completeness. The 73% reduction in processing time has direct implications for program velocity: calibration cycles that previously represented multi-day engineering commitments can be executed within a single working shift, enabling calibration teams to iterate more frequently within sprint boundaries and respond to software integration issues without introducing downstream delays. This velocity improvement changes the operational model of calibration from a milestone-driven activity to a continuous engineering function, compatible with the agile and DevOps practices increasingly adopted in SDV development organizations [10]. Abboush et al. identified similar alignment benefits when continuous integration was applied to HIL testing in automotive programs, observing that the shift from batch-mode validation to continuous validation fundamentally improves the ability to detect and resolve integration defects early [5]. The calibration automation framework extends this principle to the parameter management domain, enabling calibration to participate in the same iterative development rhythms as software engineering itself.

The 68% reduction in merge error rate carries safety implications beyond operational efficiency gains. In automotive systems governed by ISO 26262 [7], parameter values that influence safety-relevant behavior -- such as fault detection thresholds, actuator limits, and diagnostic enable conditions -- must be managed with demonstrable rigor. A merge error that propagates an incorrect threshold value into an integration build can produce a latent safety defect that escapes validation and reaches production. This represents a significant improvement over the implicit judgment applied in manual operations [4]. The traceability coverage improvement from 61% to 100% directly supports compliance evidence generation: every promoted parameter change is accompanied by a complete audit record, reducing effort required to prepare documentation packages for functional safety reviews and program milestone audits [2], [8]. A virtual testing framework such as that described by Abboush et al. [12] can be integrated downstream of the calibration automation pipeline, enabling simulation-based validation of promoted parameter changes before physical HIL testing. Table 4 summarizes framework applicability across common deployment contexts.

Table 4: Framework Applicability Matrix -- Deployment Contexts

Deployment Context	Processing Time Impact	Merge Error Reduction	Traceability Impact	CI/CD Alignment	Overall Suitability
Single-domain body control program	High	High	High	Full	Excellent
Multi-domain SDV integration	Very High	Very High	Critical	Full	Excellent
OTA update calibration cycle	High	High	High	Full	Excellent
Safety-critical params (ISO 26262)	Moderate	Critical	Critical	Partial*	Good (gate config required)
Cross-program platform reuse	High	High	Very High	Full	Excellent

[Table 4: Framework Applicability Matrix. Refer to companion workbook sheet "tables", Table 4. Columns: Deployment Context | Processing Time Impact | Merge Error Reduction | Traceability Impact | CI/CD Alignment | Overall Suitability.]

The current framework operates within defined scope boundaries that should be acknowledged. The normalization engine rule set was developed and validated against parameter encoding formats encountered in the evaluation dataset; extension to proprietary formats from additional ECU suppliers or toolchain vendors will require incremental rule development consistent with ASAM MCD-2 MC specifications [13]. The orchestration engine gate logic is configured at deployment time and requires engineering input to define domain-appropriate promotion criteria -- a deliberate design choice to preserve engineering authority over safety-critical promotion decisions, but one that introduces an initial configuration investment. Additionally, the framework evaluation was conducted on program datasets from a specific automotive domain; while the architectural principles are general, quantitative performance figures may vary across programs with significantly different parameter encoding practices or validation workflow structures [14].

7. Future Directions: Toward Autonomous Calibration Management

The clever calibration automation framework serves as the foundation of the next generation of calibration intelligence. The most pressing of these next-gen developments is the application of machine learning to the delta detection and classification engine. Beyond just sorting parameter deltas by magnitude and range of change, a machine learning-based classifier could be trained to leverage the history of how individual parameter changes affect validation metrics downstream to flag high-risk deltas sooner and help focus engineering on reviewing/validating the most important model changes. For example, Kanagaraj and Thallam [1] showed how regression models trained on historical ECU calibration trial data can be used to predict parameter value ranges in a compressed development cycle, enabling risk classification in the change management layer. Likewise, Theissler et al. see predictive analytics as a high-value extension of automated maintenance frameworks (AMFs) for automotive systems, particularly for calibrating parameters under different operating conditions [4].

Eventually, autonomous calibration management will be integrated into a digital twin environment, where the digital twin of the calibration framework can be used in a live simulation to explore how the system would behave, given proposed changes to parameters, before they go into the physical validation environment. This would reduce the amount of physical HIL testing per calibration cycle, speeding up the development cadence while retaining the safety assurance. Abboush et al. demonstrated that virtual testing frameworks built on HIL simulation principles provide high-fidelity validation coverage for automotive software parameter changes with significantly reduced physical test overhead [12]. As calibration automation matures and digital twin fidelity improves, it becomes feasible to construct calibration workflows that are fully autonomous for lower-risk parameter classes -- executing delta detection, merge, simulation-based validation, and promotion without direct engineering intervention -- while preserving human oversight for safety-critical categories under ISO 26262 [7]. This evolution positions calibration management as an adaptive, self-optimizing capability within the broader SDV engineering ecosystem [6].

Conclusion

This paper presented an intelligent calibration automation framework that transforms embedded systems calibration from a manual, engineer-dependent workflow into a governed, scalable, and reproducible infrastructure capability. The framework addresses three structural failure points of conventional calibration practice -- manual data handling, uncontrolled merging, and incomplete traceability -- through an integrated three-layer architecture comprising automated data processing pipelines [3], versioned change management systems [2], and orchestration engines [5]. Empirical evaluation across multi-domain automotive calibration datasets at production scale demonstrated a 73% reduction in processing time, a 68% reduction in parameter merge error rate, and 100% traceability coverage for promoted changes, compared to manual workflow baselines. These improvements represent a qualitative shift in the operational model of calibration, enabling automotive programs to treat parameter management as a continuous engineering activity compatible with agile development cadences [10].

The implications of this contribution extend beyond individual program efficiency. As SDV architectures continue to expand the parameter space requiring calibration management [6], [9], the sustainability of

enterprise-scale embedded systems engineering depends on the availability of automation frameworks capable of absorbing complexity growth without proportional increases in engineering resource. The framework presented here provides a replicable, extensible architecture that addresses this challenge directly. The rule-based governance mechanisms and structured audit trail capabilities establish the framework as compatible with the compliance requirements of safety-critical automotive development under ISO 26262 [7], [8].

Future extensions incorporating machine learning-based delta classification [1], [4] and digital twin integration [12] will progressively reduce engineering intervention at each calibration cycle, moving the field toward autonomous calibration management for lower-risk parameter classes. The foundation established by this framework -- governed, traceable, and continuously integrated calibration -- makes these advanced capabilities achievable without requiring fundamental architectural change. By positioning calibration as a first-class infrastructure capability rather than a peripheral engineering task, this work contributes to the long-term viability of enterprise-scale embedded systems development in an era of accelerating software complexity.

References

[1] Kanagaraj, S., & Thallam, H. S. P. G. (2022). Automation of engine ECU calibration through CAN with Python machine learning algorithms. ARAI Journal of Mobility Technology, 2(4), 327-331. https://doi.org/10.37285/ajmt.2.4.1

[2] Maro, S., Steghoefer, J.-P., & Staron, M. (2018). Software traceability in the automotive domain: Challenges and solutions. Journal of Systems and Software, 141, 85-110. https://www.sciencedirect.com/science/article/abs/pii/S0164121218300608

[3] Munappy, A., Bosch, J., & Olsson, H. H. (2020). Data pipeline management in practice: Challenges and opportunities. In Lecture Notes in Computer Science: Vol. 12562. Product-Focused Software Process Improvement (pp. 168-184). Springer. https://doi.org/10.1007/978-3-030-64148-1_11

[4] Theissler, A., Perez-Velazquez, J., Kettelgerdes, M., & Elger, G. (2021). Predictive maintenance enabled by machine learning: Use cases and challenges in the automotive industry. Reliability Engineering and System Safety, 215, 107864. https://doi.org/10.1016/j.ress.2021.107864

[5] Abboush, M., Knieke, C., & Rausch, A. (2025). Advancing real-time validation of automotive software systems via continuous integration and intelligent failure analysis. Scientific Reports, 15, 32936. https://doi.org/10.1038/s41598-025-21416-5

[6] Liu, Y., Li, H., Ding, K., & Gao, Y. (2022). Impact, challenges and prospect of software-defined vehicles. Automotive Innovation, 5(2), 180-194. https://doi.org/10.1007/s42154-022-00179-z

[7] International Organization for Standardization. (2018). ISO 26262: Road vehicles -- Functional safety (2nd ed.). ISO. https://www.iso.org/standard/68383.html

[8] Amalfitano, D., Fasolino, A. R., Tramontana, P., & Muto, E. (2019). Using tool integration for improving traceability management testing processes: An automotive industrial experience. Journal of Software: Evolution and Process, 31(2), e2171. https://doi.org/10.1002/smr.2171

[9] Lu, S., & Shi, W. (2022). Vehicle computing: Vision and challenges. Green Energy and Intelligent Transportation, 1(1), 100008. https://www.sciencedirect.com/science/article/pii/S2949715922000038

[10] Dakkak, A., Bosch, J., & Olsson, H. H. (2024). Towards AIOps-enabled services in continuously evolving software-intensive embedded systems. Journal of Software: Evolution and Process, 36(4), e2592. https://doi.org/10.1002/smr.2592

[11] Wohlrab, R., et al. (2021). Why and how your traceability should evolve: Insights from an automotive supplier. In Proceedings of the 29th IEEE International Requirements Engineering Conference (pp. 198-207). IEEE. https://ieeexplore.ieee.org/document/9425817/

[12] Abboush, M., Knieke, C., & Rausch, A. (2024). A virtual testing framework for real-time validation of automotive software systems, based on hardware-in-the-loop and fault injection. Sensors, 24(12), 3733. https://doi.org/10.3390/s24123733

[13] Association for Standardization of Automation and Measuring Systems. (2022). ASAM MCD-2 MC: Measurement and calibration data specification (Version 1.6). ASAM. https://www.asam.net/standards/detail/mcd-2-mc/

[14] Haghighatkhah, A., et al. (2017). Improving the state of automotive software engineering. IEEE Software, 34(3), 41-47. https://doi.org/10.1109/MS.2017.3571571

[15] Pretschner, A., Broy, M., Kruger, I. H., & Stauner, T. (2007). Software engineering for automotive systems: A roadmap. In Proceedings of the IEEE International Conference on Software Engineering -- Future of Software Engineering Track (pp. 55-71). IEEE. https://ieeexplore.ieee.org/document/4221612