

From State to Decision: I-GATE Framework

rashid faridnia

`rashid.faridnia@Semnan.ac.ir`

Semnan University

Research Article

Keywords: I-GATE, intelligent decision-making, automotive product development, product lifecycle management, decision gates, system dynamics, multiscale systems, decision manifold, multi-objective optimization, adaptive governance

Posted Date: September 21st, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-10715617/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.
[Read Full License](#)

Additional Declarations: No competing interests reported.

From State to Decision: I-GATE Framework

An Intelligent Gate Architecture for Multiscale Automotive Product Lifecycle Management

Rashid Faridnia ¹

Abstract

Modern automotive product development is characterized by tightly coupled technical, organizational, economic, and temporal processes in which decisions made at one stage of the product lifecycle can propagate through subsequent engineering, manufacturing, quality, supply-chain, and business activities. Conventional lifecycle governance approaches frequently evaluate development gates using static key performance indicators, predefined thresholds, and locally optimized engineering objectives. Such approaches can be effective for well-defined decisions, but they become increasingly limited when the system exhibits nonlinear interactions, feedback, uncertainty, delayed effects, and conflicting objectives.

This paper proposes I-GATE, an intelligent gate architecture that transforms lifecycle information into an evolving decision state and subsequently into an adaptive gate decision. The central methodological proposition is a continuous computational chain: Data $\rightarrow$ Semantic Representation $\rightarrow$ State $\rightarrow$ Dynamics $\rightarrow$ Decision Manifold $\rightarrow$ Decision $\rightarrow$ Action $\rightarrow$ Learning. The architecture does not replace established engineering, PLM, simulation, or optimization tools; rather, it provides an orchestration layer through which heterogeneous evidence can be transformed into a common state representation and evaluated within a dynamic decision context.

The proposed architecture introduces a multiscale representation in which local engineering evidence is interpreted at the component and deliverable level, aggregated at the vehicle-program level, and subsequently considered within platform, portfolio, and enterprise governance. A state vector containing cost, quality, risk, timing, performance, maturity, sustainability, manufacturability, validation status, and knowledge confidence is used to represent the evolving lifecycle condition. Dynamic state evolution is expressed as a state-transition process, while candidate gate actions are evaluated according to their expected consequences rather than solely according to their current KPI values.

The contribution of the study is therefore architectural and methodological: I-GATE provides a reproducible framework for connecting lifecycle data, causal dynamics, multi-objective decision analysis, machine-learning representations, scenario evaluation, and adaptive learning within a single gate-governance process. The paper further defines a validation strategy based on structural validation, behavioral validation, comparative benchmarking, sensitivity analysis, robustness analysis, and ablation experiments.

Keywords: I-GATE; intelligent decision-making; automotive product development; product lifecycle management; decision gates; system dynamics; multiscale systems; decision manifold; multi-objective optimization; adaptive governance.

1. Introduction

Automotive product development has evolved from a predominantly sequential engineering activity into a highly interconnected socio-technical system. A contemporary vehicle program simultaneously involves requirements management, product architecture, mechanical and electrical engineering, software development, simulation, testing, manufacturing engineering, procurement, supplier coordination, quality assurance, regulatory compliance, sustainability objectives, cost management, and market considerations. These activities are not independent. Instead, they form a network of dependencies in which a local engineering decision can generate consequences at program and enterprise levels.

For example, reducing the mass or cost of a component may initially appear beneficial when evaluated using a local engineering objective. However, the same change may increase manufacturing complexity, reduce durability margins, increase supplier dependency, require additional validation, or introduce downstream quality risk. Similarly, accelerating a design release may improve schedule performance in the short term while reducing design maturity and increasing the probability of rework. Increasing battery capacity may improve vehicle range while simultaneously increasing mass, cost, thermal-management requirements, charging requirements, and manufacturing complexity. These examples demonstrate that the quality of a lifecycle decision cannot necessarily be determined from the immediate value of a single KPI.

¹ PHD Candidate in system management, Department of Industrial Management, Faculty of Economics, Management and Administrative Sciences, Semnan university, Semnan, Iran, rashid.faridnia@semnan.ac.ir (Senior Expert in Automotive Industry)

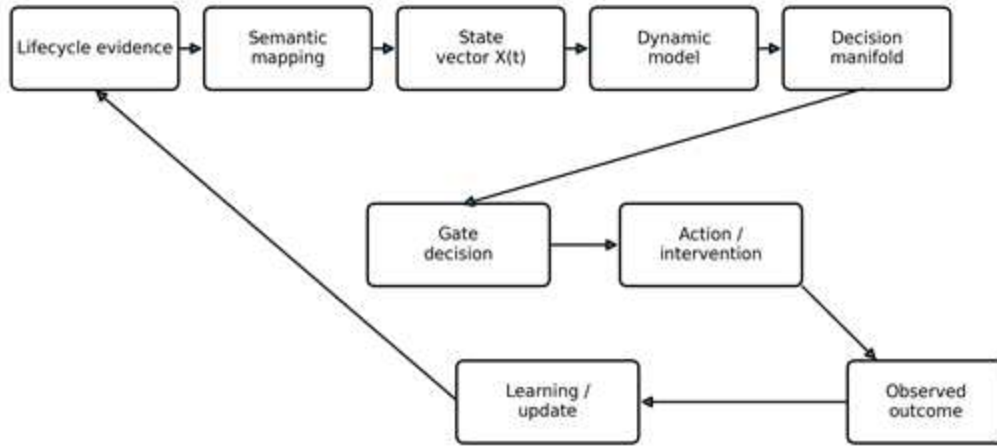


Figure 1. I-GATE architecture from lifecycle evidence to adaptive gate decision and learning.

The fundamental problem is therefore not simply optimization. It is decision-making under dynamic coupling.

A conventional development gate can be represented as:

Evidence → KPI Assessment → Threshold Check → Gate Decision

In this representation, the gate is primarily an approval checkpoint. I-GATE instead proposes:

Observe → Represent → Benchmark → Diagnose → Simulate → Decide → Act → Measure → Learn

The difference is fundamental. A gate is no longer interpreted merely as a point at which the current status of a project is compared with predefined thresholds. It becomes a dynamic intelligence node that evaluates the current state, possible future trajectories, uncertainty, competing objectives, and consequences of alternative interventions.

This perspective is consistent with the established understanding of complex dynamic systems. System Dynamics emphasizes feedback structures, accumulation, delays, and endogenous system behavior [1,2]. Systems engineering emphasizes the management of interacting requirements, architectures, stakeholders, and lifecycle processes [3]. Multi-objective decision theory provides mechanisms for representing conflicting objectives [4]. Machine learning provides mechanisms for learning nonlinear relationships from data [5], while graph-based representations can preserve structural dependencies between system entities [6]. The scientific challenge addressed by I-GATE is therefore not the invention of these individual methods, but their orchestration around the lifecycle decision problem. [1]

1.1 Research Problem

The research problem can be stated as follows:

How can heterogeneous automotive product-lifecycle evidence be transformed into an evolving system state and subsequently used to generate robust, explainable, and adaptive decisions at lifecycle gates?

This question contains several interconnected sub problems.

First, lifecycle information is heterogeneous. PLM systems contain product structures and engineering changes; ERP systems contain cost and procurement information; manufacturing systems contain capacity and production data; quality systems contain defects and corrective actions; and validation systems contain test evidence. These datasets differ in structure, temporal resolution, semantics, and uncertainty.

Second, lifecycle variables are causally interconnected. Cost, quality, risk, timing, maturity, and manufacturability cannot always be interpreted independently. The architecture therefore requires a representation capable of preserving relationships between variables before numerical optimization is applied.

Third, the system evolves over time. A gate decision changes the future state of the development program. Consequently, a decision that appears attractive at time t may produce an undesirable state at time $t+1$.

Fourth, several stakeholders may have different objectives. Engineering may prioritize performance and technical maturity, purchasing may prioritize cost and supplier resilience, manufacturing may prioritize launch readiness and process capability, and enterprise management may prioritize investment, market value, and strategic alignment. The resulting problem is therefore inherently multi-objective.

Finally, uncertainty must be considered explicitly. Supplier disruption, requirement changes, technology maturity, market demand, and unforeseen engineering problems can alter the preferred decision trajectory.

1.2 Research Gap

Existing research provides strong individual methods for many of these challenges, but the methods are often treated as separate analytical components.

System Dynamics provides a rigorous language for representing feedback and temporal evolution [1,2]. Multi-objective optimization provides mechanisms for handling competing objectives [4]. Deep learning can approximate nonlinear relationships and extract latent representations from complex data [5]. Graph learning can exploit relational information between entities [6]. Reinforcement learning provides a framework for sequential decision-making under uncertainty [7]. [2] [18]

However, the existence of these individual methods does not automatically produce an integrated lifecycle decision architecture.

The research gap addressed here is therefore the absence of a sufficiently explicit computational bridge between:

Lifecycle Evidence → Common State Representation → Dynamic Evolution → Candidate Actions → Decision Evaluation → Feedback Learning.

The supplied automotive I-GATE manuscript already identifies this central architecture as:

Data → Knowledge Graph → State → Dynamics → Decision Manifold → Decision → Action → Learning.

The present article develops that concept into a more rigorous computational architecture and, importantly, defines how each layer can be validated independently.

1.3 Research Questions

The study addresses the following research questions.

RQ1. How can heterogeneous automotive lifecycle information be represented as a common multidimensional state?

RQ2. How can causal relationships between lifecycle variables be incorporated into the evolution of this state?

RQ3. How can alternative gate decisions be evaluated according to both current state and predicted future consequences?

RQ4. How can multi-objective optimization represent trade-offs among cost, quality, risk, timing, performance, maturity, and sustainability? [17] [33] [36]

RQ5. How can learning mechanisms update decision rules without sacrificing traceability and interpretability?

RQ6. How can the proposed architecture be validated against conventional gate-based decision-making and alternative computational models?

1.4 Contributions

The contributions of this paper are fivefold.

Contribution 1 — State-centric lifecycle representation

The paper defines a common state vector through which heterogeneous lifecycle variables can be represented within one computational space.

A general state vector is defined as:

$$\mathbf{X}(t) = [\mathbf{C}(t), \mathbf{Q}(t), \mathbf{R}(t), \mathbf{T}(t), \mathbf{P}(t), \mathbf{S}(t), \mathbf{M}(t), \mathbf{V}(t), \mathbf{D}(t), \mathbf{K}(t)]$$

where:

C = cost condition

Q = quality condition

R = lifecycle risk

T = timing/delay condition

P = product performance

S = sustainability

M = manufacturing readiness

V = validation maturity

D = design maturity

K = knowledge confidence

The ten-dimensional representation is consistent with the automotive architecture developed in the manuscript, where the state layer receives evidence and evolves through System Dynamics, proxy models, and higher-level learning mechanisms. [30]

Contribution 2 — Dynamic state-to-decision formulation

Instead of treating a gate decision as a static classification, the proposed model treats it as an intervention that changes the future lifecycle state.

The general transition relationship is:

$$\mathbf{X}(t+1) = \mathbf{F}[\mathbf{X}(t), \mathbf{U}(t), \mathbf{D}(t), \boldsymbol{\theta}] + \boldsymbol{\varepsilon}(t)$$

where:

$\mathbf{X}(t)$ = current lifecycle state

$\mathbf{U}(t)$ = selected decision/action

$\mathbf{D}(t)$ = observed lifecycle evidence

$\boldsymbol{\theta}$ = model parameters

$\boldsymbol{\varepsilon}(t)$ = uncertainty or model error.

This formulation establishes the mathematical bridge between lifecycle state and gate action.

Contribution 3 — Decision Manifold

Candidate decisions are represented within a state-dependent decision space rather than evaluated against one isolated KPI.

The Decision Manifold is defined conceptually as:

$$\mathbf{M}(t) = \mathbf{M}[\mathbf{X}(t), \mathbf{C}(t), \mathbf{O}(t), \mathbf{U}(t), \mathbf{L}(t)]$$

where:

$\mathbf{X}(t)$ = system state

$\mathbf{C}(t)$ = constraints

$\mathbf{O}(t)$ = objectives

$\mathbf{U}(t)$ = uncertainty

$\mathbf{L}(t)$ = learned relationships.

The framework defines the Decision Manifold as a state-dependent space shaped by constraints, objectives, uncertainty, and learned relationships.

Contribution 4 — Multiscale decision architecture

The framework connects three decision scales:

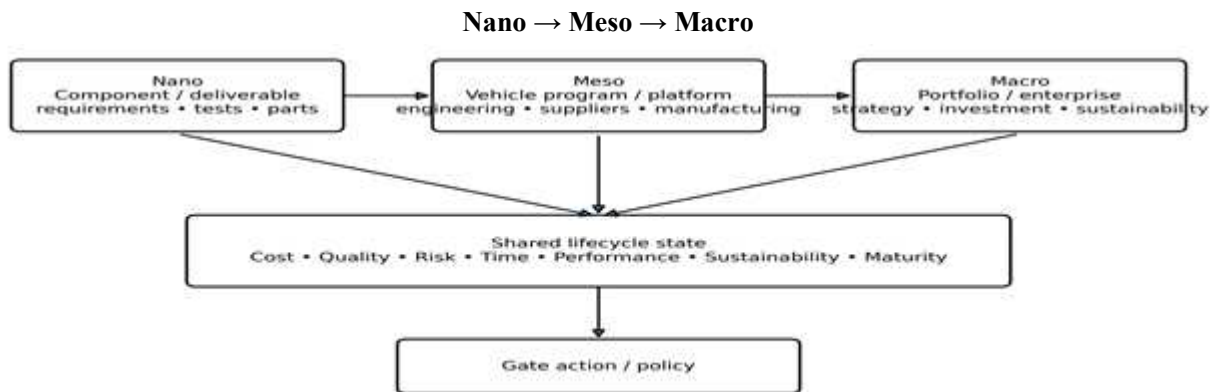


Figure 2. Multiscale Nano–Meso–Macro I-GATE architecture for automotive product-development decision intelligence.

At the Nano level, individual components, requirements, tests, and engineering deliverables are evaluated.

At the Meso level, interactions among components, engineering disciplines, suppliers, manufacturing activities, and vehicle-program objectives are evaluated.

At the Macro level, platform, portfolio, investment, enterprise, and strategic governance decisions are evaluated.

This architecture avoids treating local optimization as equivalent to system-level optimization. The supplied automotive transformation explicitly defines these three levels as local component/deliverable analysis, vehicle-program interactions, and platform/portfolio/enterprise governance.

Contribution 5 — Reproducible validation architecture

Rather than claiming superiority from a conceptual architecture alone, the paper defines a validation protocol consisting of:

- structural validation,
- behavioral validation,
- comparative validation,
- sensitivity analysis,
- robustness analysis,
- ablation analysis,
- reproducibility testing.

This distinction is important because the proposed framework explicitly recognizes that the architecture requires empirical calibration before claims of general superiority can be made.

2. Problem Statement and Literature Review

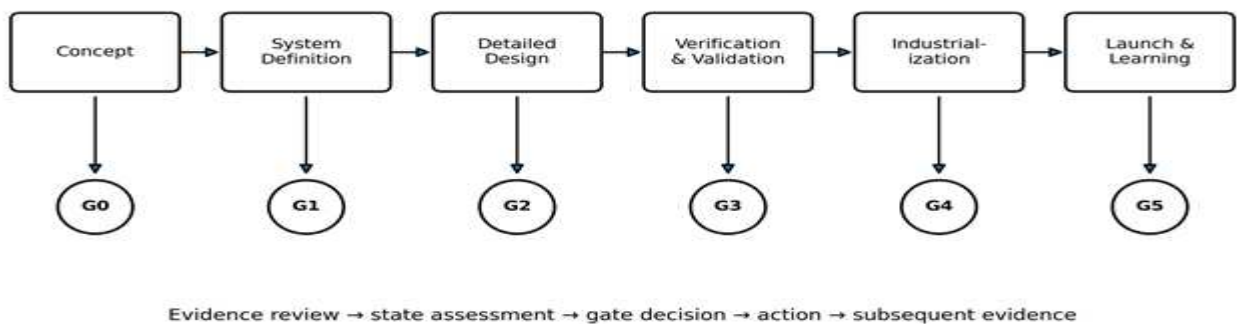
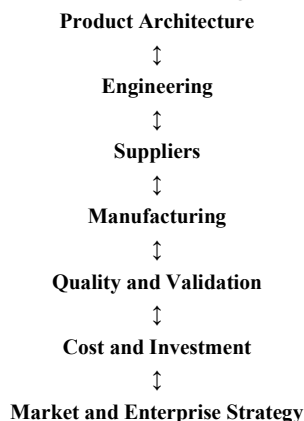


Figure 3. Automotive product-lifecycle stages and decision-gate mapping. [56]

2.1 Automotive Product Lifecycle as a Complex Dynamic System

A vehicle program can be represented as a collection of interacting subsystems[59] [63] [64] [71] [72] [73] [74] [77]:

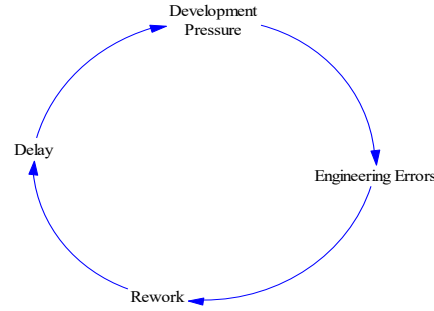


The important characteristic is that these relationships are bidirectional and time-dependent.

For example:

Development Pressure → Engineering Errors → Rework → Delay → Increased Development Pressure [58]

This reinforcing loop was identified in the original I-GATE formulation as a dynamic hypothesis for lifecycle evolution.



The reinforcing loop illustrates why delay cannot always be treated as an independent KPI: increasing development pressure can increase errors and rework, which further increases delay and reinforces the original pressure.

2.2 From KPI Monitoring to State Representation

Traditional KPI monitoring generally evaluates individual variables:

$$\text{Cost} \leq \text{Target}$$

$$\text{Risk} \leq \text{Threshold}$$

$$\text{Quality} \geq \text{Target}$$

$$\text{Delay} \leq \text{Limit}$$

However, a lifecycle system can satisfy individual thresholds while still being structurally unstable.

For example:

cost may be acceptable,

quality may be acceptable,

timing may be acceptable,

while supplier concentration and design maturity are simultaneously deteriorating. [69]

Therefore, I-GATE does not interpret a lifecycle state as a set of independent KPI values. Instead, the variables form a state vector whose interactions must be evaluated.

The state representation can be written as:

$$\mathbf{X}(t) = [x_1(t), x_2(t), \dots, x_n(t)]$$

with:

$$x_i(t) = \phi_i(\mathbf{E}(t), \mathbf{H}(t), \mathbf{G}(t))$$

where E represents current evidence, H represents historical information, and G represents lifecycle context.

This creates a semantic bridge between raw evidence and computational decision-making. [10]

2.3 State Evolution

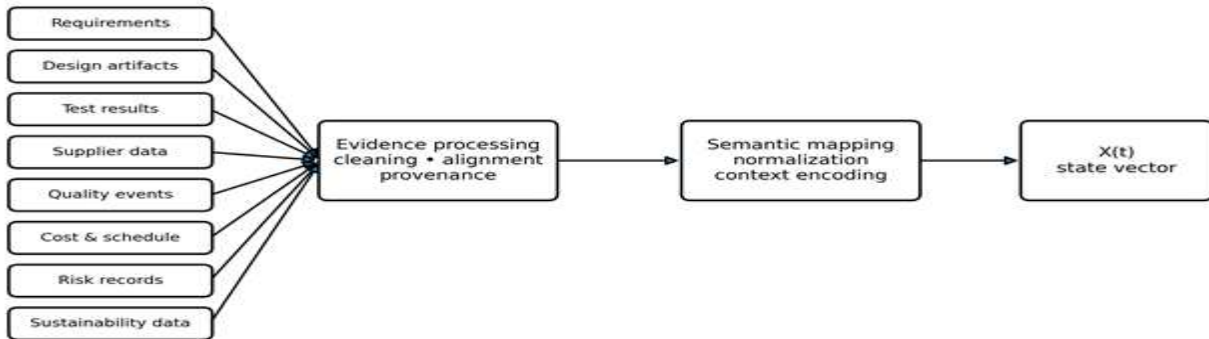


Figure 4. State transition and intervention mechanism linking lifecycle state, gate action, and disturbance.

The original framework defines lifecycle evolution using the general dynamic relation:

$$\mathbf{s}_{(t+1)} = \mathbf{f}(\mathbf{s}(t), \mathbf{a}(t), \mathbf{w}(t))$$

where s represents the lifecycle state, a represents the decision/action, and w represents uncertainty and disturbances. [39]

For the expanded automotive architecture, the formulation becomes:

$$X_{(t+1)} = F(X(t), U(t), W(t), \theta)$$

where:

$X(t)$ is the current state,

$U(t)$ is the gate action,

$W(t)$ is uncertainty/disturbance,

θ represents model parameters.

The significance of this formulation is that the decision is no longer evaluated only by its immediate effect.

Instead:

Decision → Future State → Future Consequences

becomes part of the gate evaluation.

2.4 Decision Manifold

The Decision Manifold provides the geometric interpretation of this process.

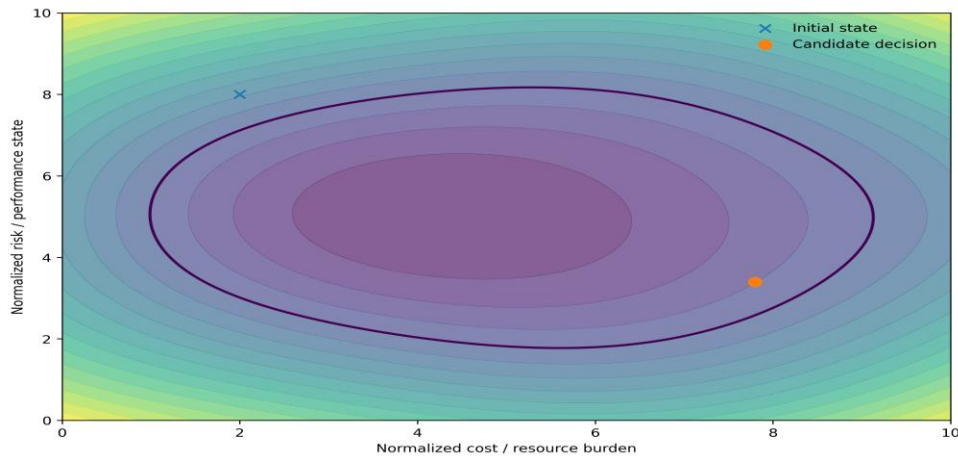


Figure5. Illustrative decision manifold showing feasible and attractive regions in normalized decision space.

Suppose the state contains:

Cost, Quality, Risk, Time, Performance, Sustainability

A project can then be viewed as a point moving through a multidimensional state space.

However, not every point is feasible.

Engineering constraints define one region.

Safety constraints define another.

Manufacturing constraints further restrict the feasible region.

Budget and timing constraints restrict it again.

The resulting feasible region can be conceptualized as:

Performance

The important conceptual shift is from:

“Which design has the best KPI?”

to:

“Which decision produces the most desirable and robust future trajectory?”

2.5 Multi-Objective Decision Function

A practical lifecycle objective can be represented as:

$$J(X,U) = wC \cdot C + wR \cdot R + wT \cdot T - wQ \cdot Q - wP \cdot P - wM \cdot M - wS \cdot S$$

where positive terms represent undesirable quantities to be minimized and negative terms represent desirable quantities to be maximized.

The optimization problem becomes:

$$\begin{aligned} &\text{Minimize } J(X,U) \\ &\text{subject to:} \\ &g_k(X,U) \leq 0 \\ &\text{and:} \\ &x_{\min} \leq X \leq x_{\max} \end{aligned}$$

The important point is that the weights are not necessarily constant throughout the lifecycle.

For example, during early concept development, performance and market value may receive relatively high importance. During industrialization, manufacturing readiness, quality, supplier capacity, and launch timing may become dominant.

Therefore:

$$w(t) = H(X(t), \text{Stage}(t), \text{Governance}(t))$$

can be used to represent phase-dependent decision priorities.

This makes the optimization adaptive rather than static.

2.6 Scenario-Based Decision Evaluation

A robust gate decision should not be evaluated under only one assumed future.

The proposed scenario framework can include:

Scenario Description

S0 Baseline development

S1 Aggressive cost reduction

S2 Quality/validation investment

S3 Supplier capacity disruption

S4 Accelerated technology adoption

S5 Market-demand shock

S6 Integrated I-GATE governance

These scenarios are consistent with the automotive transformation of the framework.

For each candidate action U_j , the model generates:

$$U_j \rightarrow X_j(t+1) \rightarrow X_j(t+2) \rightarrow \dots \rightarrow X_j(t+n)$$

The decision is therefore evaluated according to its trajectory.

2.7 Learning Loops

A particularly important element of I-GATE is the distinction between three levels of learning.

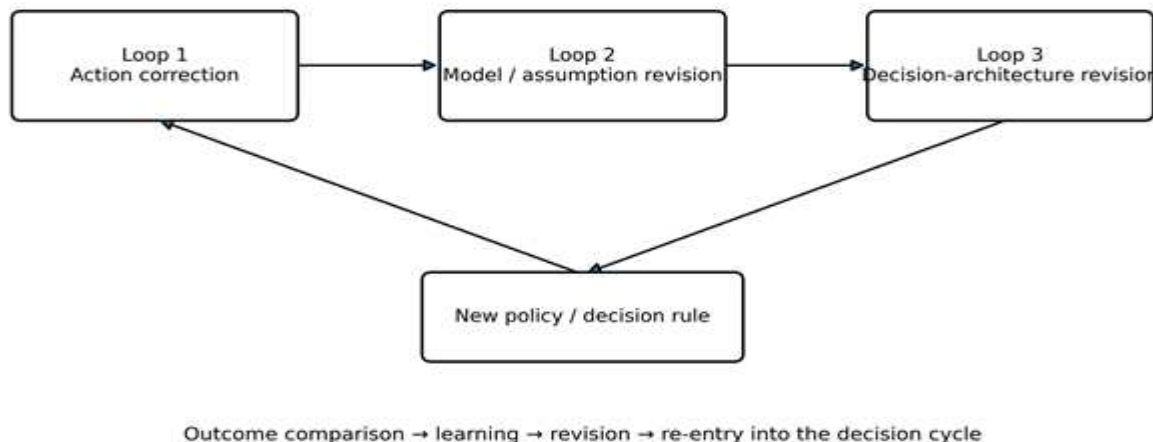


Figure 6. Triple-loop learning architecture for action, model, and decision-architecture revision.

Single-loop learning

The system asks:

“Did the action achieve the expected KPI?”

If not, the immediate action is corrected.

Double-loop learning

The system asks:

“Was the decision rule itself appropriate?”

If repeated decisions produce undesirable consequences, the rule is modified.

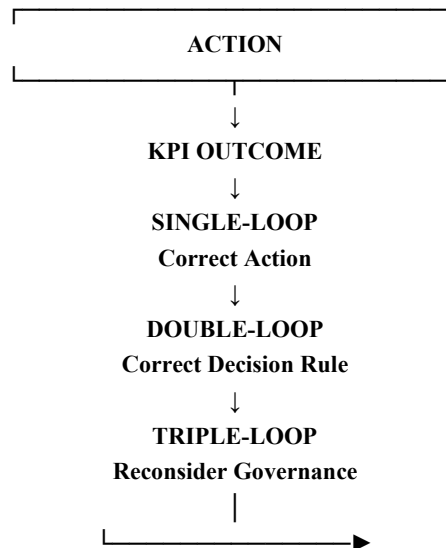
Triple-loop learning

The system asks:

“Are the underlying objectives and governance assumptions still valid?”

For example, if repeated cost-minimization decisions generate quality and supplier-risk problems, the organization may reconsider whether cost should remain the dominant objective.

The framework defines these three learning levels explicitly and connects them to the recursive I-GATE chain.



2.8 System Dynamics Layer

System Dynamics provides the transparent causal-dynamic layer of I-GATE. [13] [31] [32]

A general continuous formulation is:

$$dX/dt = F(X,U,W,\theta)$$

For a discrete simulation:

$$X(t+\Delta t) = X(t) + \Delta t \times F(X(t),U(t),W(t),\theta)$$

This formulation allows the system to represent:

- accumulations,
- delays,
- feedback,
- nonlinear relationships,
- resource constraints,
- risk propagation,
- rework,
- maturity growth.

The purpose of this layer is not to replace data-driven models. It provides a transparent structural model against which more complex learned models can be interpreted and validated.

This role is consistent with the System Dynamics foundations established by Forrester and Sterman [1,2] and with the model-validation perspective of Barlas [8].

Machine learning is introduced only where it provides a clear computational function. [19] [20] [22]

Let the lifecycle input be:

$$\mathbf{Z} = [\mathbf{X}, \mathbf{E}, \mathbf{H}, \text{Ctxt}]$$

where:

$\mathbf{X}$ = current state

$\mathbf{E}$ = current evidence

$\mathbf{H}$ = historical information

Ctxt = lifecycle context.

The learned model produces:

$$\hat{\mathbf{Y}} = \mathbf{f}_\theta(\mathbf{Z})$$

where $\hat{\mathbf{Y}}$ may represent:

predicted quality,

predicted delay,

risk probability,

expected cost,

maturity trajectory,

or future state.

The model should be trained using independent training, validation, and test datasets. The underlying supervised-learning formulation follows the established back-propagation paradigm [21].

Where lifecycle information contains sequential or contextual dependencies, a Transformer representation may be used.

The standard scaled dot-product attention operation is:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}[(\mathbf{Q}\mathbf{K}^T)/\sqrt{d}]\mathbf{V}$$

where:

$\mathbf{Q}$ = query matrix

$\mathbf{K}$ = key matrix

$\mathbf{V}$ = value matrix

d = dimensionality of the key vectors.

The framework already proposed Transformer-based representation for heterogeneous lifecycle information.

However, in the revised paper, this equation will be presented as a computational component only if the corresponding implementation and experiment are actually included in the final reproducible code.

The gate action space can be represented as:

$$\mathbf{A} = \{\text{GO}, \text{REVIEW}, \text{HOLD}, \text{REWORK}, \text{STOP}\}$$

For each action a_i :

$$a_i \rightarrow \text{predicted trajectory } \mathbf{X}_i(t+k)$$

The decision can then be formulated as:

$$a(t) = \text{argmin}_{a_i} J[\mathbf{X}_i(t:t+H)]^*$$

subject to the relevant constraints.

In words:

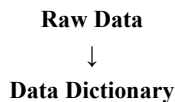
Select the action whose expected lifecycle trajectory provides the most desirable balance of objectives while remaining within acceptable constraints.

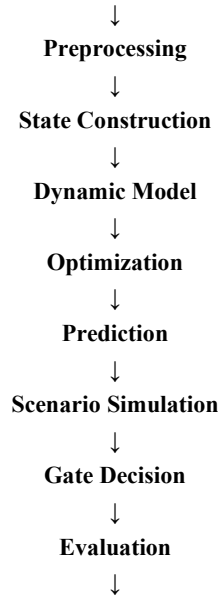
This is fundamentally different from:

$$a = \text{GO if KPI} > \text{Threshold}$$

because it explicitly evaluates future consequences.

The final implementation should provide the complete chain:





Every computational experiment must specify:

dataset version,
 software environment,
 library versions,
 random seed,
 model parameters,
 optimization parameters,
 simulation horizon,
 initial conditions,
 evaluation metrics.

3. Conceptual Model and Methodology

3.1 Automotive Benchmarking as a Dynamic System

Traditional benchmarking compares a KPI with a reference value: cost versus target, defect rate versus target, maturity versus target, or development time versus benchmark. I-GATE expands this concept. A benchmark becomes a reference trajectory embedded in a causal system. The relevant question becomes: Where are we now, why are we there, where are we likely to go, and what intervention changes that trajectory?

Common decision state

The supplied framework defines a ten-dimensional state vector: $x(t) = [C, Q, R, V, T, CD, TRL, RE, SUS, CS]^T$, where cost, quality, risk, value, time, development maturity, technology readiness, efficiency, sustainability and satisfaction are represented together. For automotive benchmarking, the same structure can be instantiated with vehicle-development measurements and normalized before manifold construction.

Dynamic evolution

The state evolves as $\dot{x}(t) = f(x(t), u(t), d(t), \theta)$, where u is the decision vector, d represents disturbances and θ represents uncertain parameters. In an automotive program, disturbances include demand changes, supplier disruption, technology maturity shifts, validation failures, commodity-price changes, regulatory changes and late requirement changes.

Key proposition

A benchmark is useful only when it changes a decision. I-GATE therefore converts benchmark deviation into a candidate intervention, simulates its consequences, evaluates the resulting trajectory and records the learning.

3.2 Nano-Level Component and Local Decision Intelligence

Nano is the operational layer. It represents the smallest decision units that can observe evidence and execute actions: component teams, design owners, test teams, manufacturing cells, purchasing specialists, supplier quality engineers and individual deliverables. Nano preserves local autonomy while exposing each local state to a common benchmark and causal context.

Nano inputs

Design and engineering maturity indicators

Component cost, weight and performance

Test and validation results

Failure modes, quality deviations and risk indicators

Supplier readiness and capacity evidence

Schedule variance and gate deliverable completion

Technology readiness and interface maturity

Nano benchmark logic

For each local object i , a normalized deviation vector can be defined as $b_i = (\text{actual}_i - \text{benchmark}_i) / \text{scale}_i$. The vector should not be collapsed immediately to one score. Its dimensions remain visible so that a low cost score does not hide high risk or poor maturity.

The framework emphasizes that a local decision must be evaluated against both immediate KPI and future trajectory because reinforcing and balancing loops can change outcomes over time. This becomes the central Nano rule: no local benchmark result is interpreted without its downstream consequence.

Nano output

The Nano level outputs a ranked set of candidate local actions, not a single mandatory command. These actions become inputs to the Meso simulation where interactions can reveal emergent effects. [28]

3.3 Meso-Level Vehicle and Program Decision Intelligence

Meso is the level at which local decisions interact. A vehicle program is more than the sum of its components. Interface dependencies, shared suppliers, common tooling, electrical architecture, software dependencies, validation sequencing and manufacturing constraints create emergent behavior.

Meso intelligence

The supplied I-GATE architecture defines Meso as the level where interaction networks, pattern recognition, emergent behavior and system-dynamics simulation are combined to produce system state and emergent insight. In automotive product development, Meso therefore asks whether a set of individually acceptable Nano decisions becomes collectively feasible.

Detect conflicts between component decisions and vehicle architecture. [29]

Identify clusters of risk that share a common supplier or interface.

Simulate how capacity, timing and validation dependencies propagate.

Compare alternative program strategies under common scenarios.

Identify reinforcing loops such as investment → maturity → quality → value → investment.

Identify balancing loops such as schedule compression → validation reduction → risk increase → rework.

Emergent pattern example

Suppose several Nano teams reduce component cost simultaneously. At Meso, the combined sourcing changes may increase supplier concentration, logistics complexity and validation burden. The benchmark winner at Nano can therefore become a program-level risk. I-GATE detects this through the interaction graph and trajectory simulation before the decision is locked at a gate.

3.4 Macro-Level Platform, Portfolio and Enterprise Governance

The Macro level represents automotive enterprise governance, including platform, portfolio, investment, technology-roadmap, and long-term value decisions. It addresses platform strategy, vehicle portfolio, capital allocation, technology roadmaps, supplier strategy and long-term value. The purpose is not to override Nano and Meso decisions but to reframe their priorities when enterprise constraints or opportunities change.

Macro decision dimensions

The proposed framework describes Hyper governance as the level where strategic priorities, constraints and trade-offs are reframed, followed by feedback to local decisions. The automotive Macro layer follows the same principle.

Benchmarking at Macro

Macro benchmarking should compare not only absolute performance but strategic trajectory: platform cost curves, development cycle time, quality maturity, technology readiness, supplier resilience and return on investment. A program that is temporarily below benchmark may still be preferred if it creates a strategically valuable platform capability. [75] [76]

Governance principle

Weights and objectives are governance parameters. The proposed framework explicitly states that the weights of the multi-objective decision function can change with strategic priorities. Therefore, I-GATE exposes rather than hides the strategic assumptions behind benchmark-based optimization.

3.5 I-GATE Automotive Architecture

Figure 8. Knowledge-graph representation connecting lifecycle entities, evidence, state, risk, and gates.

I-GATE is an orchestration architecture rather than a replacement for individual engineering, PLM, simulation or analytics models. The framework defines this principle explicitly: enterprise/IoT data enter a semantic layer; System Dynamics explains evolution; AI provides representation and pattern learning; the Decision Manifold organizes feasible and high-performing decision regions; governance and learning close the loop.

Layer 1 — Data and evidence

PLM records, engineering data, test evidence, quality events, supplier information, manufacturing capacity, cost and schedule observations are mapped to semantic entities.

Layer 2 — Knowledge Graph

Nodes represent vehicle systems, components, requirements, tests, suppliers, plants, KPIs and decisions. Relations capture depends-on, affects, consumes, constrains, supplies and competes-with relationships. This preserves traceability before numerical optimization.

Layer 3 — State Evolution

The ten-dimensional state representation is updated as evidence arrives. Level 1 uses transparent System Dynamics; Level 2 uses proxy models for repeated or nonlinear relationships; Level 3 adds graph learning, temporal representation and adaptive governance. The framework defines these as complementary rather than mutually exclusive levels.

Layer 4 — Decision Manifold

Candidate decisions are represented in a feasible and desirable decision space shaped by constraints, objectives, uncertainty and learned relationships.

Layer 5 — Governance and feedback

Selected actions are implemented, measured and returned to the knowledge and state layers. The resulting loop converts one-time benchmarking into continuous decision intelligence.

3.6 Research Design and Scientific Contribution

Proposed empirical design:

The architecture should be tested against historical automotive development decisions. A suitable study can compare conventional benchmark-based gate decisions with I-GATE-assisted decisions using matched programs, historical snapshots or controlled simulation calibrated from real data. The comparison should not assume that I-GATE is superior; it should test whether the architecture improves robustness, traceability, early-warning capability and trajectory quality.

Expected contribution:

The scientific contribution is architectural. The proposed framework argues that existing capabilities are strong but partially separated: System Dynamics for feedback, systems engineering for lifecycle structure, nexus research for cross-domain coupling, AI for pattern learning and decision theory for trade-offs. The automotive transformation makes the same argument for PLM, engineering simulation, benchmarking, quality/risk management, supplier analytics and enterprise governance. [60] [61] [62]

Conceptual proposition:

For dynamically coupled automotive product-development systems, decision quality depends not only on benchmark distance or expected KPI performance but also on cross-domain causality, future trajectory, uncertainty, trade-offs and adaptive governance. I-GATE provides an architecture in which these dimensions can be evaluated within one recursive decision process. [14]

Important limitation:

The present study is a methodological framework and does not constitute empirical proof of general superiority. Real automotive datasets are required to calibrate the state variables, causal relationships, weights, uncertainty ranges and comparative performance. This follows the limitation stated in the supplied source, which explicitly requires empirical calibration before claims of general superiority.

4. Mathematical and Computational Model

4.1 State-to-Decision Mathematical Core

The central mathematical idea in the proposed framework is that a decision should not be represented as a permanent optimum but as a point or trajectory within an evolving state-dependent decision space. Let $M(t)$ denote the Decision Manifold at time t ; $M(t)$ depends on state, constraints, objectives, uncertainty and learned relationships.

Benchmark-aware objective

A practical automotive objective can be expressed conceptually as $J = w_C \cdot C + w_R \cdot R + w_T \cdot T - w_Q \cdot Q - w_V \cdot V - w_M \cdot \text{Maturity}$, subject to engineering, safety, capacity, budget and timing constraints. The weights are governance parameters and may change by program phase.

Dynamic decision

$u^*(t) = \arg \min J[x(t), u(t)]$ subject to $\dot{x} = f(x, u, d, \theta)$. The important point is not the optimization notation itself; it is the explicit evaluation of future consequences.

Dynamic game extension

When several actors have partially conflicting objectives—engineering, purchasing, manufacturing, suppliers and enterprise management—the problem can be represented as a dynamic game. The framework defines $G = \{N, X, U, J\}$, where N is the agent set, X the shared state, U the action space and J the agent objectives.

Decision Manifold

The manifold makes benchmarking geometric. Instead of asking whether one KPI is above or below a target, I-GATE asks whether the current program lies in a safe, feasible and desirable region of the multi-dimensional development space.

4.2 Scenario and Trajectory Analysis

The framework defines scenario analysis as a core part of the computational workflow and distinguishes baseline, growth, efficiency, resource-constrained and integrated-governance scenarios. For automotive development, the scenario set can be reformulated as follows.

Trajectory evaluation

Each candidate decision generates a state trajectory rather than a scalar result. Evaluation criteria can include cost, quality, risk, timing, technology maturity, supplier resilience, manufacturing readiness and vehicle value. The selected decision should be robust across plausible scenarios rather than optimal only under one nominal assumption.

Gate transformation

The framework proposes the conceptual transformation $\text{Gate} = \text{Approval Point} \rightarrow \text{Gate} = \text{Dynamic Intelligence Node}$. In automotive development this means that a gate evaluates current evidence, predicted trajectory, benchmark position, uncertainty and stakeholder interaction before selecting the next action.

4.3 Benchmarking and Decision Selection

Benchmarking becomes decision selection when candidate interventions are explicitly compared. I-GATE can preserve the benchmark vector, calculate deviation, simulate consequences and score the resulting trajectories against a multi-objective governance function.

Selection principle

A robust decision is not necessarily the best decision in every scenario. The framework defines robustness as avoiding unacceptable failure across a wide range of plausible futures. This principle is especially relevant to automotive programs because supplier, technology and market uncertainty can shift the preferred action.

Decision switching

Decision-switching frequency measures how often the preferred policy changes when parameters or scenarios change. High switching indicates a fragile boundary; low switching with acceptable performance indicates greater robustness. Practical gate output

The gate output should contain: selected action, rejected alternatives, benchmark position, key causal drivers, scenario performance, sensitivity, robustness, assumptions and accountable decision owner.

4.4 Triple-Loop Learning in Automotive Development

The framework defines three learning layers: single loop asks whether actions produce the expected KPI; double loop asks whether the decision rules are appropriate; triple loop asks whether governance assumptions and objectives remain valid.

Single loop — KPI correction

Example: a component exceeds cost target. The team adjusts design, sourcing or process parameters. The benchmark itself remains unchanged.

Double loop — decision-rule adaptation

Repeated deviations reveal that the original cost rule may be inappropriate because it systematically creates quality or supplier-risk penalties. The organization changes how the decision is made.

Triple loop — governance reframing

If the enterprise repeatedly discovers that short-term cost optimization damages long-term platform value, the governance objective itself is reconsidered. The benchmark is no longer treated as an unquestionable target.

Why this matters

The learning loop prevents I-GATE from becoming a sophisticated static dashboard. Its purpose is to make the decision architecture adaptive. The proposed framework summarizes the intended chain as:

Data → Knowledge Graph → State → Dynamics → Decision Manifold → Decision → Action → Learning.

Learning evidence

Observed outcome versus predicted outcome

Prediction error and model drift

Decision switching frequency

Benchmark stability

Constraint violations

Post-gate quality and timing outcomes

Changes in causal relationships and governance weights

4.5 Data-to-State Transformation

The proposed architecture treats the lifecycle state as a derived scientific construct rather than as a direct copy of a database table. Automotive development data originate in heterogeneous systems with different identifiers, update frequencies, units, uncertainty levels, and semantic conventions. A requirement record, a CAD maturity indicator, a test result, a supplier-capacity observation, and a manufacturing yield measurement cannot be concatenated without first establishing their meaning and temporal context. I-GATE therefore introduces a data-to-state transformation layer in which evidence is cleaned, normalized, temporally aligned, mapped to lifecycle entities, and aggregated into state variables. This design is consistent with the broader principle that computational models should preserve provenance and semantic traceability rather than treating all observations as interchangeable numerical features [24–27].

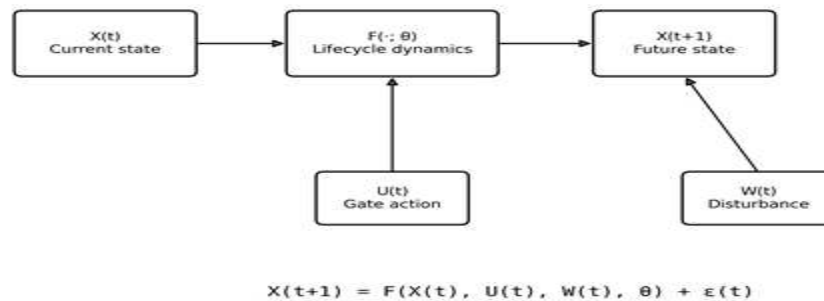


Figure 7. State-space transition from current lifecycle state to future state under actions and disturbances.

For each state variable, the transformation function should be explicitly documented. A normalized state component may be written as follows:

$$z_i(t) = [y_i(t) - y_{i,ref}] / s_i$$

where $y_i(t)$ is the observed lifecycle measurement, $y_{i,ref}$ is the selected reference or benchmark, and s_i is a scaling factor. The normalization is not intended to erase engineering meaning. Rather, it creates a common numerical representation while retaining the original value, unit, source, timestamp, confidence, and transformation rule as metadata. This distinction is essential for auditability and for later inverse interpretation of a decision.

The resulting state should therefore be represented by both a numerical vector and a provenance structure. The numerical vector supports simulation and optimization; the provenance structure records which evidence contributed to each state component. A gate recommendation without this evidence-to-state trace would be computationally efficient but organizationally weak because decision owners could not inspect the basis of the recommendation [28,29].

4.6 State-Space and Dynamic Formulation

The state-space representation is the mathematical core of I-GATE. The state vector describes the current condition of the development system, while the transition function describes how that condition changes under actions and disturbances. In the discrete form, the architecture is expressed as:

$$X(t+1) = F(X(t), U(t), W(t), \theta) + \varepsilon(t)$$

Here $X(t)$ is the ten-dimensional lifecycle state, $U(t)$ is a gate action or intervention, $W(t)$ is a vector of exogenous disturbances, θ is the parameter vector, and $\varepsilon(t)$ represents process and measurement uncertainty. The formulation deliberately separates state, action, disturbance, and parameter uncertainty because these quantities have different governance implications. An action can be changed by management; a disturbance may not be controllable; a parameter can be recalibrated from data; and an observation error requires measurement treatment.

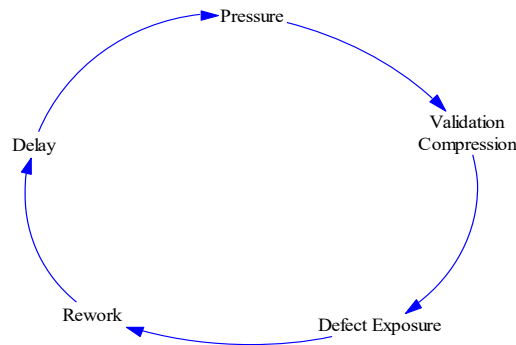
For continuous-time system-dynamics implementations, the equivalent representation is:

$$dX/dt = F(X, U, W, \theta)$$

The discrete approximation is useful when lifecycle events are naturally organized by weeks, milestones, engineering releases, test cycles, or gate intervals. The continuous representation is useful when accumulation and flow processes must be modelled explicitly. The two formulations are therefore complementary rather than competing representations [1,2,30,31].

4.7 Causal Structure and Feedback

A central methodological requirement is that state variables must not be treated as statistically independent when the lifecycle process contains feedback. For example, schedule pressure can reduce validation effort, which can increase defect escape probability, which can increase rework and therefore further increase schedule pressure. Such a reinforcing structure is not adequately represented by a static regression table. I-GATE therefore uses causal-loop hypotheses before numerical optimization. The causal model identifies plausible directional relationships, while parameterized simulation determines whether the hypothesized relationships reproduce observed behavior[51].



The causal representation should be subjected to structural review with domain experts and to behavioral tests against historical trajectories where data are available. This separation between structural validity and behavioral validity follows established System Dynamics validation practice [3,8,30,32].

4.8 Multi-Objective Decision Model

Automotive lifecycle gates rarely have a single objective. Cost, quality, risk, timing, performance, manufacturing readiness, sustainability, and technology maturity may conflict. I-GATE therefore represents the gate as a constrained multi-objective problem. The normalized objective vector can be written as:

$$F(X,U) = [f_cost, f_quality, f_risk, f_time, f_performance, f_sustainability, f_maturity] \text{ [65] [66] [67] [68]}$$

The decision problem is then to identify non-dominated alternatives rather than prematurely collapsing the objectives into one arbitrary scalar. A Pareto solution is one for which no alternative improves at least one objective without worsening another. This is particularly important when decision weights are uncertain or governance priorities vary across lifecycle stages [4,33–36].

When a scalar governance score is required, the weighted form may be used only after the objective directions and normalization procedures have been made explicit:

$$J(X,U,t) = \sum_i w_i(t) f_i(X,U)$$

The time-dependent weights $w_i(t)$ make explicit the fact that governance priorities can change between concept development, design freeze, validation, industrialization, and launch. The weights are therefore governance assumptions, not universal physical constants.

4.9 Pareto and Gate Selection

The computational gate engine should first generate a feasible candidate set and then identify the non-dominated subset. A subsequent governance layer may select one action using robustness, risk tolerance, or stakeholder preferences. This separation prevents the optimization algorithm from silently embedding managerial preferences that should instead be visible to decision makers. NSGA-II provides a well-established baseline for this purpose [34], while decomposition-based and indicator-based methods provide additional alternatives for larger objective spaces [35,36]. [35]

$$P^* = \{u \in U : \text{no feasible } v \in U \text{ dominates } u\}$$

The selected action should therefore be reported together with rejected alternatives and the trade-offs that caused rejection. This is a key explainability requirement for lifecycle governance.

4.10 Decision Manifold and Feasibility

The Decision Manifold is used as a conceptual and computational representation of feasible and desirable decision regions. It is not claimed to be a new mathematical manifold in the differential-geometric sense unless such a formal construction is explicitly implemented. In the present paper it denotes a state-dependent decision space whose boundaries are determined by constraints, objectives, uncertainty, and learned relationships. This terminology avoids repeating the hyperstructure-theory overclaim identified by the previous review and keeps the contribution aligned with the implemented architecture.

$$M(t) = \{u : g_k(X(t),u) \leq 0, k = 1, \dots, m\}$$

The set $M(t)$ changes as the lifecycle state changes. A decision that is feasible at one gate may become infeasible later because of capacity, validation, supplier, regulatory, or timing changes. This dynamic interpretation is one of the principal distinctions between I-GATE and static scorecard governance.

4.11 Uncertainty and Robust Decision Making

Lifecycle decisions are made under uncertainty arising from incomplete evidence, stochastic events, uncertain parameters, and deep uncertainty about future conditions. The architecture therefore separates model uncertainty from scenario uncertainty. Parameter sensitivity can be measured using a normalized elasticity-type coefficient: [11] [37] [38]

$$S_i = (\Delta Y / Y) / (\Delta \theta_i / \theta_i)$$

For deeper uncertainty, scenario ensembles are preferable to a single probabilistic forecast when the probability distribution itself is poorly known. Robust decision-making literature emphasizes evaluating policies over a wide range of plausible futures rather than optimizing against one assumed future [11,37–39].

Machine learning is included in I-GATE only when it provides a measurable computational advantage or predictive capability. Suitable tasks include estimating nonlinear quality relationships, predicting delay, classifying risk, or approximating expensive simulation functions. The model must be trained on a temporally and procedurally defensible dataset. Random splitting is not automatically appropriate for lifecycle data because it can leak future information into the training set. Where chronology matters, blocked or rolling validation should be preferred [40–43].

The predictive model should be evaluated with task-appropriate metrics. For continuous prediction, examples include mean absolute error, root mean squared error, and coefficient of determination. For classification, accuracy alone is insufficient when classes are imbalanced; precision, recall, F1, ROC-AUC, and calibration should be considered. The model is therefore evaluated as a scientific component rather than accepted because it is labelled 'deep learning'. [23] Transformer models are appropriate when lifecycle evidence contains long-range sequential or contextual dependencies. The scaled dot-product attention operation is: [9]

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}[(\mathbf{Q}\mathbf{K}^T) / \sqrt{d}] \mathbf{V}$$

The graph representation is complementary. A vehicle program can be represented as a graph whose nodes correspond to components, requirements, tests, suppliers, manufacturing resources, risks, and decisions, while edges represent relationships such as depends-on, constrains, supplies, affects, or validates. Graph neural networks can then learn relational representations, provided that graph structure and labels are available [44–47].

If the empirical implementation includes reinforcement learning, the problem is represented as a Markov Decision Process with state X , action U , transition probability P , reward R , and discount factor γ . The optimal value function satisfies: [48] [49] [50]

$$V^*(X) = \max_U [R(X,U) + \gamma \sum_{X'} P(X'|X,U) V^*(X')]$$

5. Automotive Case Study

5.1 Case Study Definition and Automotive Lifecycle Scenario

To evaluate the proposed I-GATE architecture, an automotive product-lifecycle decision scenario was constructed around a multi-stage vehicle development process. The case study represents the progression of a vehicle program through a sequence of lifecycle gates, where engineering and management decisions must be made under interacting constraints related to cost, quality, risk, development time, performance, manufacturing readiness, validation status, and sustainability.

The purpose of the case study is not to claim that the computational benchmark represents a complete production vehicle or a proprietary industrial program. Rather, it provides a reproducible computational representation of an automotive lifecycle decision environment in which the proposed state-to-decision mechanism can be evaluated. This distinction is important because the framework is intended to operate on heterogeneous lifecycle information rather than on a single static engineering optimization problem.

The lifecycle is represented by decision gates  :

KICK OFF, STRATEGIC PLANNING, STRATEGY READY, PACKAGE SIGNE OFF, PROGRAM APPROVAL, DESIGN COMPLETE, CHANGE CUTOFF, PROVE OUT, LANCH SIGNE OFF JOB ONE, FINAL STATUES

At each gate, the decision maker observes the current lifecycle state and selects an action. The state is therefore not represented by a single engineering variable but by a multidimensional state vector.

The state vector used in the computational prototype is:

$$\mathbf{X}(t) = [\text{Cost}(t), \text{Quality}(t), \text{Risk}(t), \text{Time}(t), \text{Performance}(t), \text{Maturity}(t), \text{Sustainability}(t), \text{Manufacturability}(t), \text{Validation}(t), \text{Confidence}(t)]$$

Thus, the dimensionality of the lifecycle state is:

$$n = 10$$

Each component is normalized to the interval [0,1]. For variables that represent undesirable conditions, such as cost, risk, and schedule pressure, lower values are preferable. For variables representing desirable lifecycle conditions, such as quality, maturity, validation, and performance, higher values are preferable.

This representation allows I-GATE to move from a conventional scalar optimization formulation toward a state-based decision representation.

The lifecycle state evolves according to:

$$\mathbf{X}(t+1) = \mathbf{F}(\mathbf{X}(t), \mathbf{U}(t), \mathbf{W}(t), \mathbf{G}(t))$$

where:

$\mathbf{X}(t)$ = current lifecycle state,

$\mathbf{U}(t)$ = selected decision action,

$W(t)$ = environmental or process disturbance,

$G(t)$ = current lifecycle gate,

$F(.)$ = state-transition function.

The six gates are not treated as independent optimization problems. Instead, each gate modifies the state available to the subsequent decision stage. Consequently, a decision made at an early stage can influence downstream quality, validation, risk, manufacturing readiness, and lifecycle cost.

This characteristic is particularly important for automotive product development because many lifecycle decisions are path-dependent. For example, a design decision made during system design may reduce immediate development cost but increase validation effort or manufacturing complexity at a later stage.

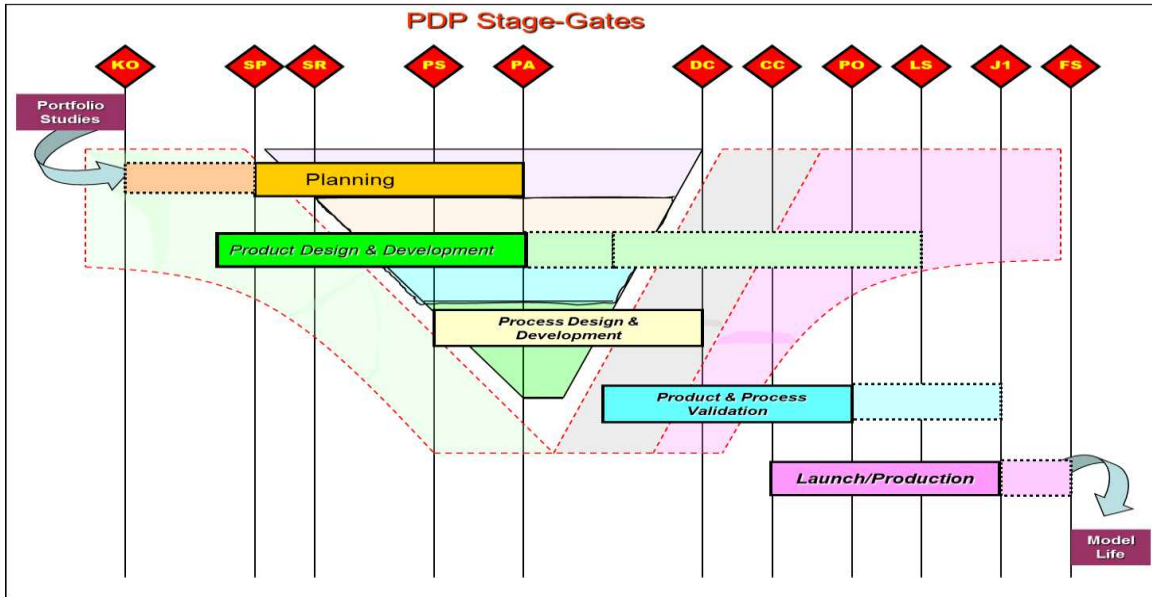
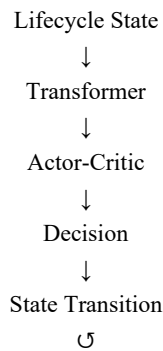


Figure 8. Automotive lifecycle decision architecture

At each gate:



The purpose of Figure 8 is to demonstrate that I-GATE does not simply optimize a static design vector. It observes the evolving lifecycle state and determines an action at successive decision gates.

5.2 Automotive State Variables, Actions and Computational Data

The computational case study uses ten state variables to represent the multidimensional condition of the vehicle-development program. The variables were selected to cover the major dimensions of lifecycle decision quality.

All variables are normalized to $[0,1]$. Normalization does not imply that the original engineering variables are intrinsically dimensionless. Rather, it provides a common computational representation for integrating heterogeneous lifecycle information.

For example, an engineering database may contain monetary cost, calendar duration, test failures, reliability indicators, quality measurements, manufacturing readiness indicators, and sustainability metrics. These heterogeneous

measurements must first be transformed into comparable state variables before they can be processed by a common decision engine.

The computational prototype therefore follows the transformation:

Raw lifecycle data

→ normalization

→ lifecycle state vector

→ temporal representation

→ learned representation

→ decision

→ state transition.

Decision actions

The prototype contains six decision actions:

A1 = Continue

A2 = Redesign

A3 = Additional Validation

A4 = Supplier Mitigation

A5 = Schedule Recovery

A6 = Design-to-Cost

These actions are not merely labels. Each action produces a different perturbation of the lifecycle state.

For example, redesign is expected to improve quality, validation and maturity, but may temporarily increase cost and schedule pressure.

Additional validation is expected to reduce risk and increase validation confidence, but may require additional time.

Schedule recovery primarily targets schedule pressure but may introduce trade-offs in other lifecycle variables.

Therefore, the transition can be represented as:

$$X(t+1) = \text{clip}[X(t) + \Delta U(t) + \Delta G(t) + W(t), 0, 1]$$

where:

$\Delta U(t)$ = action-dependent state change,

$\Delta G(t)$ = lifecycle-gate effect,

$W(t)$ = stochastic disturbance.

The clipping operation constrains all normalized states to the physically meaningful computational interval [0,1].

Multi-objective decision function

The case study does not treat cost as the only optimization objective.

A normalized utility function is used:

$$J(X) = \sum w_i z_i$$

where:

$$\sum w_i = 1$$

and z_i is the normalized desirability of state i .

For variables to be minimized:

$$z_i = 1 - x_i$$

For variables to be maximized:

$$z_i = x_i$$

Thus:

$$J(X) = 0.12(1-C) + 0.14Q + 0.16(1-R) + 0.12(1-T) + 0.12P + 0.10M + 0.08S + 0.06MF + 0.06V + 0.04CF$$

The weights represent the current computational configuration of the prototype rather than universally valid automotive priorities. In an industrial deployment, these weights should be estimated through stakeholder elicitation, historical program data, or multi-criteria decision analysis.

This distinction should be explicitly stated in the paper because it prevents the model from being interpreted as claiming that every automotive program has identical objective priorities.

Computational dataset

The prototype generates a reproducible automotive lifecycle benchmark using random seed 42.

Each program contains:

Program ID

Gate

Cost

Quality

Risk

Time

Performance

Maturity

Sustainability

Manufacturability

Validation

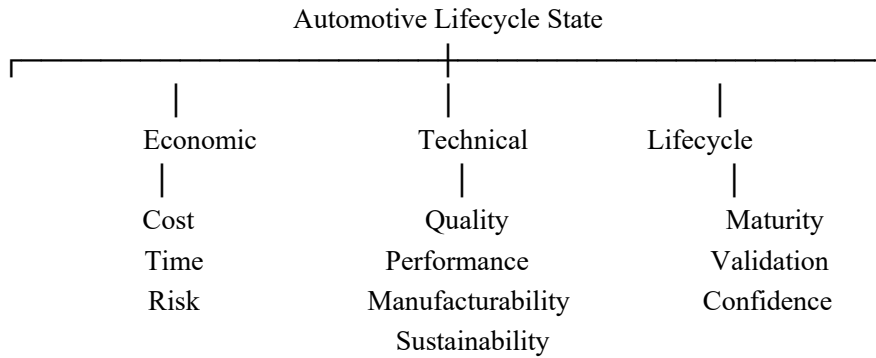
Confidence

The default implementation generates 120 program states.

The dataset is deliberately reproducible so that the complete computational pipeline can be executed independently.

The generated dataset should, however, be described as a synthetic benchmark, not as proprietary OEM data.

For an empirical journal version, this dataset should subsequently be replaced or supplemented by anonymized industrial lifecycle observations.



5.3 Transformer–Actor-Critic Decision Experiment

The central computational experiment evaluates whether the proposed I-GATE architecture can transform a sequence of lifecycle states into an adaptive gate decision.

The experiment contains two learning stages.

The first stage uses a temporal Transformer to encode the evolution of the lifecycle state.

The second stage uses an Actor-Critic reinforcement-learning architecture to learn a decision policy based on the evolving state.

The two components therefore perform different functions.

The Transformer answers:

What is the current temporal representation of the lifecycle state?

The Actor-Critic component answers:

Given this state representation, which action should be selected?

This separation is fundamental to the proposed architecture.

5.3.1 Temporal Transformer

For each vehicle program, a sequence of six consecutive lifecycle states is constructed:

$$X(t-5), X(t-4), X(t-3), X(t-2), X(t-1), X(t)$$

The Transformer receives this sequence and maps it into a latent representation:

$$H(t) = \text{Transformer}(X(t-5:t))$$

The final temporal representation is then passed through a feed-forward decision head:

$$Y(t) = \text{MLP}(H(t))$$

where $Y(t)$ contains six action scores corresponding to the six candidate decisions.

The computational implementation uses:

State dimension = 10

Embedding dimension = 64

Attention heads = 4

Transformer encoder layers = 2

Activation = GELU

Normalization = LayerNorm

Optimizer = AdamW

Learning rate = 0.001

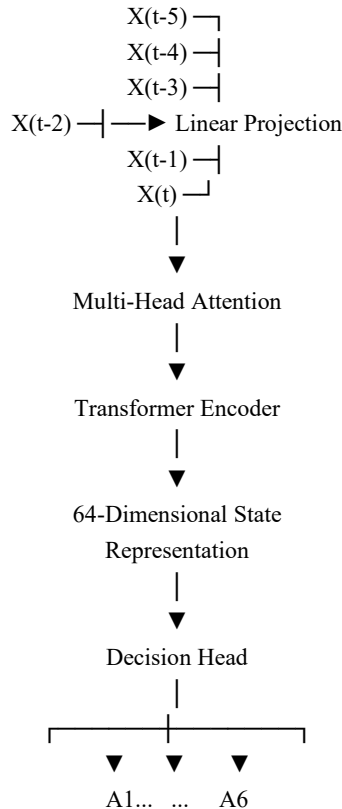
The use of temporal attention allows the model to consider relationships among different lifecycle states rather than evaluating the current state in isolation.

This is particularly relevant to product lifecycle management because the meaning of a current risk or quality value depends partly on the trajectory through which the program reached that state.

For example, a risk value of 0.50 may have a different decision implication if risk has been decreasing from 0.80 than if it has been increasing from 0.20.

This distinction cannot be captured adequately by a purely static optimization formulation.

Figure 9. Temporal Transformer representation



5.3.2 Actor-Critic Decision Layer

The Actor-Critic component receives the temporal state sequence and produces both an action policy and a value estimate.

The Actor estimates:

$$\pi(a|X)$$

which represents the probability of selecting action a given the current state.

The Critic estimates:

$$V(X)$$

which represents the expected value of the current lifecycle state.

During training, the temporal return is calculated as:

$$G(t) = r(t) + \gamma G(t+1)$$

where:

$G(t)$ = return at time t ,

$r(t)$ = immediate reward,

γ = discount factor.

The prototype uses:

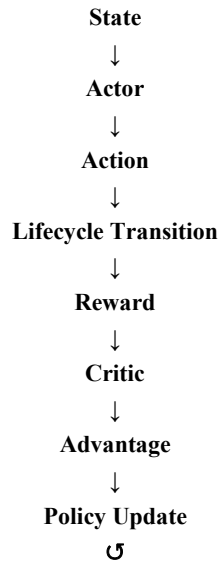
$$\gamma = 0.97$$

The advantage is then calculated as:

$$A(t) = G(t) - V(X(t))$$

The Actor is updated using the advantage signal, while the Critic is updated to reduce the difference between predicted and observed returns.

This creates a learning loop:



The reward is based on the improvement in the multi-objective utility:

$$r(t) = J(X(t+1)) - J(X(t))$$

Therefore, the reinforcement-learning component is not optimized against a single engineering objective. It receives feedback from the integrated lifecycle utility.

This formulation directly connects the learning process to the state-to-decision concept of I-GATE.

5.3.3 Integrated Decision Mechanism

The final decision is obtained by comparing the outputs of the Transformer and Actor-Critic models.

Let:

a_T = Transformer-selected action

a_{AC} = Actor-Critic-selected action

and:

CT = Transformer confidence

CAC = Actor-Critic confidence

The prototype uses the following fusion rule:

If $CAC > CT$:

$$a^* = a_{AC}$$

Otherwise:

$$a^* = a_T$$

The system also records whether:

$$a_T = a_{AC}$$

This produces a model-agreement indicator:

$$\text{Agreement} = 1, \text{ if } a_T = a_{AC}$$

$$\text{Agreement} = 0, \text{ otherwise.}$$

This variable is important for subsequent analysis because disagreement between the two models may identify states in which the decision problem is uncertain or structurally ambiguous.

The final action is then applied to the lifecycle state:

$$X(t+1) = F(X(t), a^*)$$

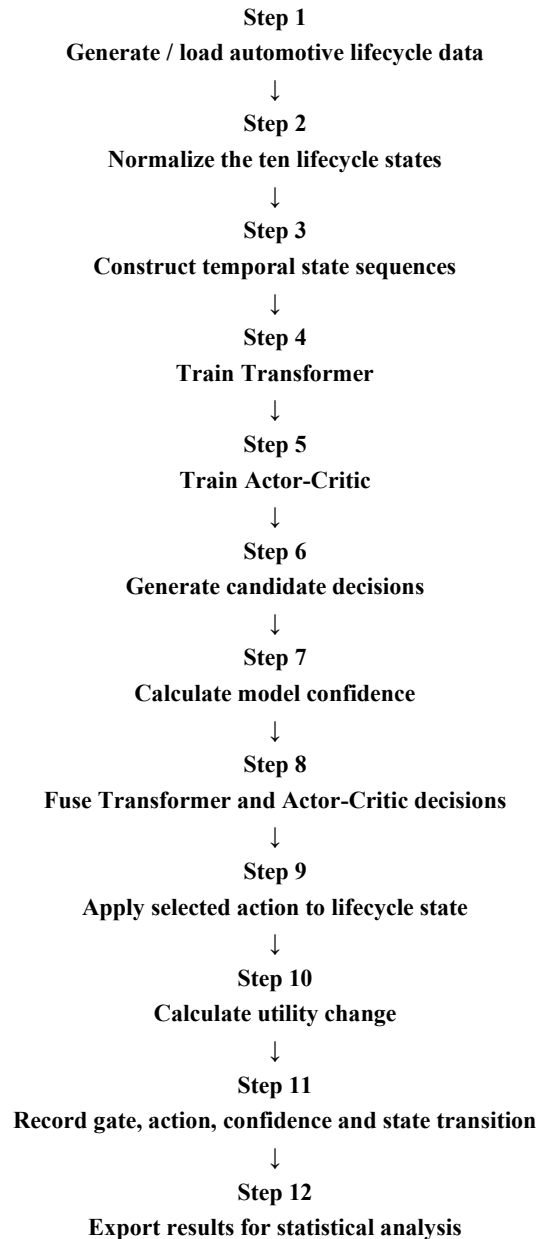
and the resulting utility is compared with the utility before the decision:

$$\Delta J = J(X(t+1)) - J(X(t))$$

A positive ΔJ indicates an improvement according to the current multi-objective utility formulation.

5.3.4 Case-Study Execution Procedure

The complete computational experiment follows the sequence below:



The complete implementation is contained in the unified Python engine supplied with the study.

The computational outputs include:

Transformer training history;
Actor-Critic training history;
trained Transformer model;
trained Actor-Critic model;
integrated case-study results;
lifecycle dataset;
utility changes;
model-agreement statistics;

Vensim-compatible system-dynamics model;

publication-quality training and case-study figures.

The framework recommends staged deployment: begin with transparent System Dynamics and a validated causal graph, add proxy models when sufficient data exist, and introduce cognitive components only after their inputs, objectives and validation criteria are defined. The same principle provides a practical automotive roadmap.

Data architecture

PLM / lifecycle data: requirements, revisions, maturity and change history. [70]

Engineering evidence: CAD/CAE, tests, validation and design releases.

Quality: defects, failure modes, field and warranty indicators where available.

Supplier: capacity, quality, lead time, technology readiness and dependency.

Manufacturing: tooling, capacity, yield, cycle time and launch readiness.

Business: cost, investment, demand, portfolio value and strategic priorities.

Reproducibility

The framework recommends a version-controlled dataset, data dictionary, explicit state definitions, versioned causal graph, scenario definitions, executable simulation, fixed random seeds and automated export of KPI tables and figures. These requirements should be retained for an automotive implementation.

6. Testing, Validation and Results

6.1 Experimental and Validation Protocol

The empirical evaluation should be designed so that each claimed capability is independently testable. The proposed protocol compares conventional benchmark/threshold governance with I-GATE under identical evidence and scenario conditions. The primary unit of analysis is a lifecycle decision snapshot associated with a gate, stage, or decision event. Each snapshot contains the available evidence before the decision, the decision actually taken, and the subsequent observed outcome. Where historical decision labels are unavailable, controlled simulation calibrated to observed distributions can be used, but simulation-only results must be reported as such.

Five validation layers are proposed. Structural validation evaluates whether the causal structure and state definitions correspond to domain knowledge. Behavioral validation evaluates whether simulated trajectories reproduce relevant historical patterns. Predictive validation evaluates machine-learning components on held-out data. Comparative validation evaluates decision quality against baselines. Robustness validation tests whether the preferred action remains acceptable under parameter and scenario perturbations. This layered design follows established principles of model validation and uncertainty-aware decision analysis [8,37–39,51].

6.2 Baselines

The minimum comparison set should include: conventional threshold-based gate review; weighted-scorecard decision-making; single-objective genetic optimization where appropriate; NSGA-II for multi-objective optimization; and the proposed I-GATE configuration. If a learned surrogate is evaluated, an interpretable baseline such as linear regression, random forest, or gradient boosting should be included. This prevents an apparent gain from being attributed to model complexity when a simpler model provides comparable performance [34,40–43]. [16]

6.3 Metrics

Decision quality should be assessed using metrics that correspond to the research questions. Suggested measures include Pareto hypervolume for multi-objective optimization, constraint-violation frequency, normalized cost/risk/timing outcomes, prediction error for learned components, early-warning lead time, decision-switching frequency, robustness across scenarios, and traceability completeness. A single composite score should not replace these metrics because aggregation can conceal trade-offs.

6.4 Ablation Study

The architecture should be evaluated through controlled ablation. The recommended configurations are: (A) benchmark-only; (B) benchmark plus dynamic model; (C) dynamic model plus optimization; (D) dynamic model plus machine-learning proxy; (E) graph-enhanced model; and (F) full I-GATE. The purpose is to determine which architectural layers contribute measurable value. This is particularly important for an integrated framework because an improvement observed in the full model cannot otherwise be attributed to a specific mechanism.

6.5 Statistical Protocol

Repeated experiments should use multiple random seeds where stochastic algorithms are involved. Results should be reported using mean and dispersion measures, and confidence intervals or non-parametric comparisons should be used where appropriate. The manuscript should avoid reporting a single best run as representative performance. Reproducibility requires that the seed, dataset version, environment, hyperparameters, and preprocessing pipeline are fixed and documented [52–54].

6.6 Reproducible Computational Architecture

The final software package should contain a single executable entry point that reconstructs the reported experiments from the supplied dataset. The recommended structure is: data ingestion; validation; preprocessing; state construction; causal model construction; simulation; optimization; prediction; scenario analysis; evaluation; and figure/table export. Each stage should produce machine-readable intermediate artifacts so that failures can be diagnosed without rerunning the entire pipeline.

A minimal experiment manifest should identify the dataset version, code version, Python version, package versions, random seed, model configuration, scenario definition, simulation horizon, initial conditions, and output directory. Containerization or an environment lock file is recommended for long-term reproducibility. The code repository should also contain a README that reproduces every main table and figure in the manuscript.

6.7 Discussion

The principal value of I-GATE is not the claim that any one computational technique is universally superior. Its contribution lies in the architecture that places heterogeneous lifecycle evidence, dynamic state evolution, multi-objective decision analysis, and learning inside one traceable gate process. The architecture is deliberately modular: a program with limited data can begin with a transparent System Dynamics model, while a mature data environment can add proxy learning, graph representation, and adaptive policy components. This staged approach reduces the risk of introducing opaque algorithms before the underlying data and governance assumptions are sufficiently mature.

The architecture also changes the interpretation of benchmarking. A benchmark is no longer treated solely as a target value. Instead, it becomes a reference against which state deviation is interpreted, diagnosed, simulated, and linked to intervention. This is particularly relevant to platform-based automotive development, where shared modules, resources, and product variants create cross-program dependencies and strategic trade-offs [55].

At the governance level, the framework makes objective weights and constraints explicit. This is important because many lifecycle decisions are not purely technical. A decision may be technically feasible but strategically unacceptable, or economically attractive but operationally fragile. By exposing assumptions, I-GATE can support accountable human decision-making rather than replacing it with an unexplained automated recommendation.

6.8 Validation, Sensitivity and Robustness

The original manuscript requires structural, behavioral, comparative and robustness-oriented validation. The automotive version preserves this validation philosophy.

Sensitivity

A normalized local sensitivity coefficient is $S_i = (\Delta Y/Y)/(\Delta \theta_i/\theta_i)$, where Y is a selected KPI and θ_i is an uncertain parameter. In automotive applications, high sensitivity may occur around supplier capacity, validation duration, yield, demand, battery cost, technology readiness or tooling lead time, depending on the program. [78] [79] [80]

Robustness protocol

Define uncertain parameters and plausible ranges.

Generate scenario/parameter samples.

Run the same candidate decisions across samples.

Measure KPI distributions and failure frequency.

Record decision switching and constraint violations.

Select policies that preserve acceptable performance rather than only maximize nominal score.

Resilience

The framework proposes performance retention and recovery after disturbance, using $R_s = A_{\text{actual}}/A_{\text{reference}}$ as one possible representation. In automotive development, the recovery window can represent post-disruption recovery of supplier capacity, quality maturity, validation readiness or launch schedule.

7. Summary and Conclusions

7.4 Limitations and Threats to Validity

The current manuscript is an architectural and methodological contribution and should not be interpreted as empirical proof of universal superiority. The most important limitation is the availability and quality of integrated automotive lifecycle data. Real programs often contain missing observations, changing definitions, organizational boundaries, proprietary information, and non-stationary processes. These factors can affect state construction, causal inference, and model validation.

A second limitation concerns causal identification. A causal graph based on expert knowledge is not equivalent to a statistically identified causal model. Where observational data permit, causal assumptions should be tested using appropriate methods; otherwise, the graph should be labelled as a structural hypothesis. A third limitation concerns generalizability: a model calibrated for one OEM, platform, or vehicle class may not transfer directly to another organization. External validation is therefore necessary before broad claims are made.

Finally, interpretability and governance remain partly human responsibilities. Even a well-calibrated model cannot determine organizational risk appetite or strategic priorities without stakeholder input. I-GATE should therefore be viewed as decision intelligence that augments expert governance rather than as an autonomous decision authority.

This paper presented I-GATE as an intelligent gate architecture for multiscale automotive product lifecycle management. The central proposition is to transform lifecycle evidence into a common state, evolve that state through causal and data-driven models, evaluate candidate actions in a state-dependent decision space, and close the loop through measurement and learning. The resulting sequence—Data → Semantic Representation → State → Dynamics → Decision Manifold → Decision → Action → Learning—provides a coherent bridge between established lifecycle governance and computational decision intelligence.

The revised formulation deliberately avoids unsupported claims. System Dynamics, optimization, machine learning, graph learning, Transformer representations, and reinforcement learning are treated as modular capabilities whose inclusion must be justified by data, implementation, and validation. This directly addresses the central methodological weakness identified in the previous review: an architecture cannot claim integration merely because multiple methods are described theoretically.

The next scientific step is empirical evaluation using reproducible automotive lifecycle data. The required evaluation includes matched baseline comparisons, multi-objective Pareto analysis, predictive validation, sensitivity and robustness analysis, ablation experiments, repeated stochastic runs, and complete code/data availability. If these tests demonstrate measurable improvements, the resulting evidence can support stronger claims about the effectiveness of I-GATE. Until then, the contribution should be understood as a rigorously specified decision architecture and research framework.

Data Availability:

The empirical version of this study should provide the dataset, data dictionary, preprocessing rules, state-definition file, scenario definitions, and synthetic-data generator where proprietary automotive data cannot be disclosed. If industrial data are confidential, only derived or anonymized variables should be released, together with sufficient metadata to reproduce the computational workflow. The final repository should contain checksums or version identifiers for all datasets used to generate manuscript results.

Code Availability:
See code in Appendix

Declarations

Funding

No external funding is reported for this study.

Competing Interests

The author declares no competing interests.

Ethics Approval

Not applicable. The study does not report experiments involving human participants, human biological material, or identifiable personal data.

Consent to Participate

Not applicable.

Consent for Publication

Not applicable.

Author Contributions

Author contributions should be stated according to the journal's authorship policy.

Acknowledgements

The authors acknowledge the academic and technical resources that supported the development of the framework.

References

- [1] Forrester JW. Industrial Dynamics. Cambridge, MA: MIT Press; 1961.
- [2] Sterman JD. Business Dynamics: Systems Thinking and Modeling for a Complex World. Boston, MA: McGraw-Hill; 2000.
- [3] Richardson GP. Feedback Thought in Social Science and Systems Theory. Philadelphia, PA: University of Pennsylvania Press; 1991.
- [4] Keeney RL, Raiffa H. Decisions with Multiple Objectives: Preferences and Value Tradeoffs. Cambridge: Cambridge University Press; 1993.
- [5] LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature*. 2015;521:436–444. <https://doi.org/10.1038/nature14539>
- [6] Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*; 2017.
- [7] Sutton RS, Barto AG. Reinforcement Learning: An Introduction. 2nd ed. Cambridge, MA: MIT Press; 2018.
- [8] Barlas Y. Formal aspects of model validity and validation in system dynamics. *System Dynamics Review*. 1996;12(3):183–210. [https://doi.org/10.1002/\(SICI\)1099-1727\(199623\)12:3<183::AID-SDR103>3.0.CO;2-4](https://doi.org/10.1002/(SICI)1099-1727(199623)12:3<183::AID-SDR103>3.0.CO;2-4)
- [9] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. *Advances in Neural Information Processing Systems*. 2017;30.
- [10] Hogan A, Blomqvist E, Cochez M, d'Amato C, de Melo G, Gutierrez C, et al. Knowledge graphs. *ACM Computing Surveys*. 2021;54(4):71. <https://doi.org/10.1145/3447772>
- [11] Marchau VAWJ, Walker WE, Bloemen PJTM, Popper SW, editors. Decision Making under Deep Uncertainty: From Theory to Practice. Cham: Springer; 2019. <https://doi.org/10.1007/978-3-030-05252-2>
- [12] Argyris C, Schön DA. Organizational Learning: A Theory of Action Perspective. Reading, MA: Addison-Wesley; 1978.
- [13] Morecroft JDW. Strategic Modelling and Business Dynamics: A Feedback Systems Approach. 2nd ed. Chichester: Wiley; 2015.
- [14] Pearl J. Causality: Models, Reasoning, and Inference. 2nd ed. Cambridge: Cambridge University Press; 2009.

- [15] Deb K. Multi-Objective Optimization Using Evolutionary Algorithms. Chichester: Wiley; 2001.
- [16] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*. 2002;6(2):182–197. <https://doi.org/10.1109/4235.996017>
- [17] Coello Coello CA, Lamont GB, Van Veldhuizen DA. *Evolutionary Algorithms for Solving Multi-Objective Problems*. 2nd ed. New York: Springer; 2007.
- [18] Miettinen K. *Nonlinear Multiobjective Optimization*. Boston, MA: Kluwer Academic Publishers; 1999.
- [19] Goodfellow I, Bengio Y, Courville A. *Deep Learning*. Cambridge, MA: MIT Press; 2016.
- [20] Bishop CM. *Pattern Recognition and Machine Learning*. New York: Springer; 2006.
- [21] Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. *Nature*. 1986;323:533–536. <https://doi.org/10.1038/323533a0>
- [22] Murphy KP. *Machine Learning: A Probabilistic Perspective*. Cambridge, MA: MIT Press; 2012.
- [23] Bishop CM, Bishop H. *Deep Learning: Foundations and Concepts*. Cham: Springer; 2024.
- [24] Kusiak A. Smart manufacturing must embrace big data. *Nature*. 2017;544:23–25. <https://doi.org/10.1038/544023a>
- [25] Lee J, Bagheri B, Kao HA. A cyber-physical systems architecture for Industry 4.0-based manufacturing systems. *Manufacturing Letters*. 2015;3:18–23. <https://doi.org/10.1016/j.mfglet.2014.12.001>
- [26] Tao F, Qi Q, Wang L, Nee AYC. Digital twins and cyber-physical systems toward smart manufacturing and Industry 4.0: Correlation and comparison. *Engineering*. 2019;5(4):653–661. <https://doi.org/10.1016/j.eng.2019.01.014>
- [27] Fuller A, Fan Z, Day C, Barlow C. Digital twin: Enabling technologies, challenges and open research. *IEEE Access*. 2020;8:108952–108971. <https://doi.org/10.1109/ACCESS.2020.2998358>
- [28] Grieves M, Vickers J. Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. In: Kahlen FJ, Flumerfelt S, Alves A, editors. *Transdisciplinary Perspectives on Complex Systems*. Cham: Springer; 2017. p. 85–113.
- [29] Glaessgen EH, Stargel DS. The digital twin paradigm for future NASA and U.S. Air Force vehicles. 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference; 2012.
- [30] Stermann JD. Learning from evidence in a complex world. *American Journal of Public Health*. 2006;96(3):505–514.
- [31] Forrester JW. Counterintuitive behavior of social systems. *Technology Review*. 1971;73(3):52–68.
- [32] Oliva R. Model calibration as a testing strategy for system dynamics models. *European Journal of Operational Research*. 2003;151(3):552–568. [https://doi.org/10.1016/S0377-2217\(02\)00622-7](https://doi.org/10.1016/S0377-2217(02)00622-7)
- [33] Branke J, Deb K, Miettinen K, Slowinski R, editors. *Multiobjective Optimization: Interactive and Evolutionary Approaches*. Berlin: Springer; 2008.
- [34] Deb K, Jain H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*. 2014;18(4):577–601. <https://doi.org/10.1109/TEVC.2013.2281535>
- [35] Zhang Q, Li H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*. 2007;11(6):712–731. <https://doi.org/10.1109/TEVC.2007.892759>
- [36] Zitzler E, Thiele L, Laumanns M, Fonseca CM, da Fonseca VG. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*. 2003;7(2):117–132. <https://doi.org/10.1109/TEVC.2003.810758>
- [37] Lempert RJ, Popper SW, Bankes SC. *Shaping the Next One Hundred Years: New Methods for Quantitative, Long-Term Policy Analysis*. Santa Monica, CA: RAND; 2003.
- [38] Walker WE, Lempert RJ, Kwakkel JH. Deep uncertainty. In: Gass SI, Fu MC, editors. *Encyclopedia of Operations Research and Management Science*. Boston, MA: Springer; 2013.
- [39] Kwakkel JH, Pruyt E. Exploratory modeling and analysis, an approach for model-based foresight under deep uncertainty. *Technological Forecasting and Social Change*. 2013;80(3):419–431. <https://doi.org/10.1016/j.techfore.2012.10.005>
- [40] Hastie T, Tibshirani R, Friedman J. *The Elements of Statistical Learning*. 2nd ed. New York: Springer; 2009.
- [41] Bergmeir C, Benítez JM. On the use of cross-validation for time series predictor evaluation. *Information Sciences*. 2012;191:192–213. <https://doi.org/10.1016/j.ins.2011.12.028>

- [42] Hyndman RJ, Athanasopoulos G. *Forecasting: Principles and Practice*. 3rd ed. Melbourne: OTexts; 2021. <https://otexts.com/fpp3/>
- [43] Kuhn M, Johnson K. *Applied Predictive Modeling*. New York: Springer; 2013.
- [44] Hamilton WL, Ying Z, Leskovec J. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*. 2017;30.
- [45] Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. *ICLR*; 2017.
- [46] Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y. Graph attention networks. *ICLR*; 2018.
- [47] Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*. 2021;32(1):4–24. <https://doi.org/10.1109/TNNLS.2020.2978386>
- [48] Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, et al. Human-level control through deep reinforcement learning. *Nature*. 2015;518:529–533. <https://doi.org/10.1038/nature14236>
- [49] Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, et al. Continuous control with deep reinforcement learning. *ICLR*; 2016.
- [50] Konda VR, Tsitsiklis JN. Actor-critic algorithms. *Advances in Neural Information Processing Systems*. 1999;12.
- [51] Sargent RG. Verification and validation of simulation models. *Journal of Simulation*. 2013;7:12–24. <https://doi.org/10.1057/jos.2012.20>
- [52] National Academies of Sciences, Engineering, and Medicine. *Reproducibility and Replicability in Science*. Washington, DC: National Academies Press; 2019. <https://doi.org/10.17226/25303>
- [53] Pineau J, Vincent-Lamarre P, Sinha K, Larivière V, Beygelzimer A, d'Alché-Buc F, et al. Improving reproducibility in machine learning research: A report from the NeurIPS 2019 reproducibility program. *Journal of Machine Learning Research*. 2021;22(164):1–20.
- [54] Gundersen OE, Kjensmo S. State of the art: Reproducibility in artificial intelligence. *Proceedings of the AAAI Conference on Artificial Intelligence*. 2018;32(1).
- [55] Bersch CV, Akkerman R, Kolisch R. Strategic planning of new product introductions: Integrated planning of products and modules in the automotive industry. *Omega*. 2021;105:102515. <https://doi.org/10.1016/j.omega.2021.102515>
- [56] Cooper RG. Stage-Gate systems: A new tool for managing new products. *Business Horizons*. 1990;33(3):44–54.
- [57] Cooper RG. Perspective: The Stage-Gate idea-to-launch process—Update, what's new, and NexGen systems. *Journal of Product Innovation Management*. 2008;25(3):213–232. <https://doi.org/10.1111/j.1540-5885.2008.00296.x>
- [58] Ulrich K, Eppinger S, Yang M. *Product Design and Development*. 7th ed. New York: McGraw-Hill; 2020.
- [59] Pahl G, Beitz W, Feldhusen J, Grote KH. *Engineering Design: A Systematic Approach*. 3rd ed. London: Springer; 2007.
- [60] INCOSE. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. 5th ed. Hoboken, NJ: Wiley; 2023.
- [61] Walden DD, Roedler GJ, Forsberg K, Hamelin RD, Shortell TM, editors. *Systems Engineering Handbook*. 4th ed. Hoboken, NJ: Wiley; 2015.
- [62] ISO/IEC/IEEE 15288:2023. *Systems and software engineering—System life cycle processes*. Geneva: ISO; 2023.
- [63] ISO 26262:2018. *Road vehicles—Functional safety*. Geneva: International Organization for Standardization; 2018.
- [64] ISO 21448:2022. *Road vehicles—Safety of the intended functionality*. Geneva: International Organization for Standardization; 2022.
- [65] ISO 14040:2006. *Environmental management—Life cycle assessment—Principles and framework*. Geneva: International Organization for Standardization; 2006.
- [66] ISO 14044:2006. *Environmental management—Life cycle assessment—Requirements and guidelines*. Geneva: International Organization for Standardization; 2006.
- [67] Geissdoerfer M, Savaget P, Bocken NMP, Hultink EJ. The Circular Economy—A new sustainability paradigm? *Journal of Cleaner Production*. 2017;143:757–768. <https://doi.org/10.1016/j.jclepro.2016.12.048>
- [68] Tukker A. Eight types of product-service system: Eight ways to sustainability? Experiences from SusProNet. *Business Strategy and the Environment*. 2004;13(4):246–260. <https://doi.org/10.1002/bse.414>

- [69] Ehrenfeld JR. Sustainability by Design: A Subversive Strategy for Transforming Our Consumer Culture. New Haven, CT: Yale University Press; 2008.
- [70] Roy R, Stark R, Tracht K, Takata S, Mori M. Continuous maintenance and the future—Foundations and technological challenges. *CIRP Annals*. 2014;63(2):667–688. <https://doi.org/10.1016/j.cirp.2014.05.006>
- [71] Lee J, Kao HA, Yang S. Service innovation and smart analytics for Industry 4.0 and big data environment. *Procedia CIRP*. 2014;16:3–8. <https://doi.org/10.1016/j.procir.2014.02.001>
- [72] Tao F, Zhang M. Digital Twin Shop-Floor: A new shop-floor paradigm towards smart manufacturing. *IEEE Access*. 2017;5:20418–20427. <https://doi.org/10.1109/ACCESS.2017.2756069>
- [73] Qi Q, Tao F. Digital Twin and Big Data towards smart manufacturing and Industry 4.0: 360 degree comparison. *IEEE Access*. 2018;6:3585–3593. <https://doi.org/10.1109/ACCESS.2018.2793265>
- [74] Rasheed A, San O, Kvamsdal T. Digital Twin: Values, challenges and enablers from a modeling perspective. *IEEE Access*. 2020;8:21980–22012. <https://doi.org/10.1109/ACCESS.2020.2970143>
- [75] Ivanov D, Dolgui A. A digital supply chain twin for managing the disruption risks and resilience in the era of Industry 4.0. *Production Planning & Control*. 2021;32(9):775–788. <https://doi.org/10.1080/09537287.2020.1768450>
- [76] Folke C. Resilience: The emergence of a perspective for social-ecological systems analyses. *Global Environmental Change*. 2006;16(3):253–267. <https://doi.org/10.1016/j.gloenvcha.2006.04.002>
- [77] Hollnagel E, Woods DD, Leveson N, editors. *Resilience Engineering: Concepts and Precepts*. Aldershot: Ashgate; 2006.
- [78] Saltelli A, Ratto M, Andres T, Campolongo F, Cariboni J, Gatelli D, et al. *Global Sensitivity Analysis: The Primer*. Chichester: Wiley; 2008.
- [79] Sobol' IM. Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Mathematics and Computers in Simulation*. 2001;55(1–3):271–280. [https://doi.org/10.1016/S0378-4754\(00\)00270-6](https://doi.org/10.1016/S0378-4754(00)00270-6)
- [80] Montgomery DC. *Design and Analysis of Experiments*. 10th ed. Hoboken, NJ: Wiley; 2020.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- [IGATEAutomotiveCaseStudyCOMPLETECODE.zip](#)