SAFER v3.0: Definitive System Architecture and Design Specification for Deterministic Software-in-the-Loop Prognostics

1. Executive Summary and Strategic Imperative

The aerospace propulsion sector currently stands at a critical inflection point, caught between the escalating complexity of high-bypass turbofan engines and the rigid, deterministic safety standards that govern avionics certification. As propulsion systems evolve toward higher efficiency and tighter integration, the traditional paradigm of Scheduled Maintenance—based on fixed flight-hour intervals—is becoming increasingly untenable. It is inefficient, resulting in the premature removal of healthy components, yet paradoxically prone to missed failures in engines that degrade non-linearly. This inefficiency has driven the industry toward Condition-Based Maintenance (CBM), a paradigm predicated on the ability to accurately estimate the Remaining Useful Life (RUL) of an asset based on real-time sensor data.

However, the transition to CBM is stalled by an "Architectural Gap." While the research community has produced high-accuracy prognostic algorithms using Deep Learning, these models (Transformers, CNNs) are typically "black boxes" that lack the interpretability, determinism, and computational efficiency required for deployment in safety-critical Software-in-the-Loop (SIL) environments.

This report presents the comprehensive, ready-to-implement system design for **SAFER v3.0 (Safety-Aware Flexible Emulation for Reliability)**. This architecture is expressly engineered to bridge the gap between advanced stochastic modeling and deterministic safety assurance. Drawing on a first-principles analysis of thermodynamics and control theory, SAFER v3.0 introduces a tri-partite architecture:

1. **The Prognostic Core:** A **Mamba-based Selective State Space Model (SSM)** that replaces the computationally heavy Transformer, offering linear scalability ($O(L)$) and constant-time inference ($O(1)$) to process full flight histories without truncation.[1]
2. **The Physics Monitor:** An **Adaptive Linear Parameter-Varying (LPV) SINDy** subsystem. This module resolves the "Healthy vs. Dying" conflict inherent in static identification by treating degradation as a scheduling parameter, enabling the physics model to evolve alongside the engine.[1]
3. **The Simulation Fabric:** A **Shared-Memory Multiprocessing** transport layer that eliminates fragile C-based dependencies, employing lock-free ring buffers to ensure

deterministic data handling on standard computational hardware.[1]

This document serves as the final technical specification for SAFER v3.0, detailing the mathematical formulations, software engineering patterns, and verification logic necessary for immediate implementation.

---

2. Introduction: The Deterministic Prognostics Challenge

2.1 The Operational Context: C-MAPSS and the Data Deluge

The development of SAFER v3.0 is grounded in the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS), the NASA-developed benchmark that models a large-scale turbofan engine (90,000 lb thrust class). Modern engines are instrumented with hundreds of sensors—temperatures, pressures, rotor speeds—generating terabytes of data per flight. The C-MAPSS dataset captures the degradation of critical components, specifically the High-Pressure Turbine (HPT) and Low-Pressure Turbine (LPT), across thousands of run-to-failure trajectories.[1]

The challenge is not merely to predict failure but to do so within the constraints of an avionics environment. Avionics software must be **deterministic**: given the same input, it must produce the same output within a guaranteed timeframe. Standard deep learning models, particularly those relying on massive attention mechanisms or stochastic sampling, often violate these constraints. Furthermore, they are computationally intensive, requiring GPUs that are rarely available in flight control computers or portable ground support equipment.

2.2 The Legacy of Failure: SAFER v1.0 and v2.0

To understand the necessity of the v3.0 design, we must rigorously analyze the failures of its predecessors. These were not coding errors but fundamental theoretical misalignments.

SAFER v1.0: The Hamiltonian Fallacy
Version 1.0 attempted to enforce physics compliance using Physics-Informed Hamiltonian Neural Networks (PIHE). Hamiltonian mechanics is a powerful framework for conservative systems (e.g., planetary orbits, frictionless pendulums) where the total energy $\mathcal{H}$ is conserved ($\frac{d\mathcal{H}}{dt} = 0$).
However, a turbofan engine is a heat engine. Its very purpose is to convert chemical energy into work and heat. It is governed by the Second Law of Thermodynamics: entropy increases, and energy is dissipated.

- **Friction:** Bearings and gears convert kinetic energy into heat.
- **Irreversibility:** The combustion process is irreversible.
- Degradation: Wear, erosion, and fouling represent a permanent, dissipative loss of

component efficiency.

When a Hamiltonian network was forced to model this dissipative C-MAPSS data, it struggled to reconcile the observed energy loss with its built-in conservation constraints. The model was observed to "hallucinate" oscillatory energy exchanges to balance the equation, resulting in non-physical RUL predictions.1 Insight: You cannot model a dying engine with the math of an eternal pendulum.

SAFER v2.0: The Stationarity Fallacy

Version 2.0 utilized Sparse Identification of Nonlinear Dynamics (SINDy) to discover the differential equations of the engine. It assumed a static relationship: $\dot{x} = \Xi \Theta(x)$. This ignored the control system's response to degradation. In a healthy engine, a specific fuel flow $W_f$ produces a specific rotor speed $N_f$. In a degraded engine, the Full Authority Digital Engine Control (FADEC) detects the loss of efficiency and increases $W_f$ to maintain $N_f$.

A static SINDy model sees these two operating points (Healthy vs. Degraded) as contradictory data. It converges to an "average" model that fits neither state well, leading to high residuals throughout the flight envelope. This rendered the monitor useless for distinguishing sensor faults from genuine degradation.1 Insight: The physics of a machine change as the machine dies.

2.3 The SAFER v3.0 Mandate

SAFER v3.0 is designed to resolve these specific issues through first-principles reasoning:

1. **Embrace Dissipation:** Use State Space Models (SSMs) that naturally handle dissipative dynamics (eigenvalues within the unit circle) rather than Hamiltonian conservation.
2. **Embrace Non-Stationarity:** Use Linear Parameter-Varying (LPV) formulations to explicitly model the shift in physics as a function of health.
3. **Ensure Determinism:** Use lock-free shared memory and constant-time inference algorithms to guarantee real-time performance.

---

3. Theoretical Foundations: First-Principles Reasoning

Before detailing the software implementation, we must establish the mathematical validity of the chosen architecture. This section provides the rigorous justification for replacing Transformers with Mamba and static SINDy with LPV-SINDy.

3.1 The Sequence Length Bottleneck and the Case for SSMs

Flight data is fundamentally a long-range sequence. A degradation signature—such as a subtle shift in the Exhaust Gas Temperature (EGT) peak during takeoff—may occur hundreds of flights (tens of thousands of time steps) before the catastrophic failure.

- **The Transformer Limit:** The Transformer architecture, the current state-of-the-art in many fields, relies on the Self-Attention mechanism. Attention computes the pairwise

correlation between every time step and every other time step. This operation scales quadratically with sequence length: $O(L^2)$.
  - For a short sentence ($L=100$), this is negligible.
  - For a C-MAPSS trajectory ($L=3000+$), this is prohibitive.
  - **Consequence:** Engineers are forced to "truncate" the history, feeding only the last 50 or 100 cycles to the model. This discards the early-life history, which contains the "initial conditions" of the degradation path.
- **The Mamba Solution:** SAFER v3.0 adopts the Mamba architecture, a class of Selective State Space Models. Mamba achieves the "Holy Grail" of sequence modeling:
  1. **Linear Scaling $O(L)$:** Like an RNN, it processes data sequentially, meaning memory usage grows linearly with flight duration.
  2. **Parallel Training:** Unlike an RNN, it uses an associative scan algorithm (Section 5.3) to train in parallel, utilizing modern multi-core CPUs/GPUs efficiently.
  3. **Constant Inference $O(1)$:** During the SIL simulation, it acts as a recurrent system, requiring fixed memory regardless of how long the simulation runs.

3.2 Thermodynamics and LPV Systems

The decision to use LPV-SINDy is rooted in the thermodynamics of the Brayton Cycle. The efficiency of the cycle depends on the component efficiencies ($\eta_{compressor}, \eta_{turbine}$).

$$\text{Thrust} \propto f(W_f, P_2, T_2, \eta_{HPT}, \eta_{LPT})$$

Degradation is effectively a time-varying reduction in $\eta$. As $\eta$ decreases, the FADEC alters the fuel schedule. Therefore, the system matrices governing the linearized dynamics $\dot{x} = Ax + Bu$ are not constant ($A, B$), but are functions of the health parameter: $A(\eta), B(\eta)$.
This is the definition of a Linear Parameter-Varying (LPV) system. By using LPV mathematics, we align the monitor with the physical reality: the governing equations of the engine smoothly morph from the "Healthy Equations" to the "Broken Equations" as the degradation parameter evolves.

---

4. Prognostic Core: Mamba Architecture Specification

The prognostic core is the predictive engine of SAFER v3.0. It is responsible for ingesting the multivariate sensor stream and outputting the estimated RUL.

4.1 Architecture Overview

The core is designed as a deep sequence model. Unlike generic implementations, the SAFER v3.0 Mamba block is tuned for the specific spectral properties of mechanical vibration and thermodynamic transients found in C-MAPSS data.

Component	Specification	Description
Input Dimension	$D_{in} = 14$	The subset of C-MAPSS sensors containing prognostic value (T24, T30, T50, P30, Nf, Nc, Ps30, phi, etc.).[1]
Model Dimension	$D_{model} = 64$	The size of the latent embedding space. A dimension of 64 is sufficient to capture the correlations without overfitting.
State Dimension	$D_{state} = 16$	The expansion factor of the SSM. The effective "memory" capacity is $D_{model} \times D_{state} = 1024$ floating point values.
Layers	$N = 4$	Number of stacked Mamba blocks.
Normalization	RMSNorm	Root Mean Square Normalization is used for training stability.
Discretization	ZOH	Zero-Order Hold method for continuous-to-discrete conversion.

4.2 The Selective State Space Mechanism

The defining innovation of Mamba is the Selective SSM. Standard SSMs (like the Kalman Filter's internal model) use constant matrices A, B, C.

$$h'(t) = \mathbf{A}h(t) + \mathbf{B}x(t)$$

$$y(t) = \mathbf{C}h(t)$$

In SAFER v3.0, these matrices are functions of the input $x(t)$.

$$\Delta = \text{Softplus}(\text{Linear}(x_t))$$

$$\mathbf{B} = \text{Linear}(x_t)$$

$$\mathbf{C} = \text{Linear}(x_t)$$

Operational Insight:
This input-dependence allows the model to perform "Content-Aware Reasoning."

1. **Noise Gating:** During flight phases where sensor data is noisy or irrelevant (e.g., idling on the tarmac), the model can drive the time-step Δ toward zero. This effectively "freezes" the hidden state $h(t)$, protecting the long-term memory of degradation from being corrupted by transient noise.
2. **Event Focus:** When a significant event occurs (e.g., a rapid throttle transient), the model can increase Δ, forcing a significant update to the state vector to capture the new dynamic response.

4.3 Discretization: The Zero-Order Hold (ZOH)

The continuous equations must be discretized for digital implementation. We employ the Zero-Order Hold (ZOH) assumption, which posits that the input signal $x(t)$ is held constant during the sampling interval Δ. This is the standard assumption in digital control avionics.[1]
The discrete system matrices are computed as:

$$\bar{A} = \exp(\Delta \mathbf{A})$$

$$\bar{B} = (\Delta \mathbf{A})^{-1}(\exp(\Delta \mathbf{A}) - I) \cdot (\Delta \mathbf{B})$$

This transformation allows the model to learn in continuous time (conceptually) while executing in discrete time steps, making it robust to irregular sampling rates (e.g., if data drops out and the effective Δ changes).

4.4 Pure PyTorch Implementation for Hardware Independence

A strict requirement for SAFER v3.0 is SIL portability. Most high-performance Mamba implementations rely on NVIDIA's Triton compiler or custom CUDA kernels, which bind the software to specific GPUs.
Design Decision: SAFER v3.0 mandates a Pure PyTorch implementation.

The Parallel Scan Algorithm:
To achieve fast training without CUDA kernels, we implement the parallel associative scan using PyTorch's native tensor operations. The recurrence $h_t = \bar{A}_t h_{t-1} + \bar{B}_t x_t$ can be parallelized.
Let the operator $\otimes$ denote the combination of state transitions. The state at time $t$ depends on the cumulative product of $\bar{A}$ matrices.
$$ h_t = \left(\prod_{i=0}^t \bar{A}i \right) h{-1} + \sum_{k=0}^t \left(\prod_{j=k+1}^t \bar{A}_j \right) \bar{B}_k x_k $$
This looks computationally expensive ($O(L^2)$) if computed naively. However, because matrix multiplication is associative, we can use the Parallel Scan (or Prefix Sum) algorithm to compute it in $O(\log L)$ time steps on a parallel processor.
We implement this in PyTorch using torch.einsum for efficient batch matrix multiplication and torch.cumsum in log-space for numerical stability (preventing underflow when multiplying many small probabilities).
Inference Optimization:
During the SIL simulation loop, training speed is irrelevant; inference latency is king. Here, the Mamba model switches to its Recurrent Mode.

$$h_{t} = \bar{A}_{t-1} h_{t-1} + \bar{B}_{t-1} x_t$$

$$y_t = \bar{C}_t h_t$$

This requires only matrix-vector multiplication. The computational cost is constant $O(1)$ per time step. This guarantees that the RUL estimator will never exceed the 20ms simulation budget, regardless of whether the flight is at minute 1 or hour 10. This determinism is the critical enabler for avionics certification.

---

5. Physics Monitor: The Adaptive LPV-SINDy Subsystem

While the Mamba core provides the prediction, the LPV-SINDy monitor provides the *explanation* and the *safeguard*. It verifies that the sensor data is consistent with physical laws.

5.1 The "Healthy vs. Dying" Conflict

As established, a static physics model fails because the engine's behavior changes with age.

- **Healthy Engine:** High efficiency, lower Exhaust Gas Temperature (EGT) for a given thrust.
- Dying Engine: Low efficiency, higher EGT for the same thrust (due to FADEC compensation).

A static model trained on both will view the "Dying" data points as outliers or noise compared to the "Healthy" baseline.

5.2 LPV Formulation and Library Augmentation

SAFER v3.0 solves this by introducing a Scheduling Parameter $p(t)$ that serves as a proxy for engine health. We select the EGT Margin as this parameter.1
$$p(t) = \text{Normalized EGT Margin} \in $$

- $p=1$: Perfectly Healthy.
- $p=0$: Failure Threshold.

We reformulate the system dynamics as an LPV system:

$$\dot{x}(t) = \Xi(p(t)) \cdot \Theta(x(t))$$

We approximate the dependence of Ξ on p as a linear polynomial:

$$\Xi(p) = \Xi_0 + p \cdot \Xi_1$$

- Ξ_0: The baseline physics matrix (representing the $p=0$ failed state).
- Ξ_1: The deviation matrix (representing the "health bonus" physics).

Augmented Library Construction:
The SINDy library $\Theta(x)$ is augmented to include interaction terms between the state x and the parameter p.
Original Library: $\Theta_{base} =^T$
Augmented Library: $\Theta_{LPV} =^T$

$$\Theta_{LPV} =^T$$
When we run the sparse regression (STLSQ or Lasso) on this augmented library, the algorithm learns to assign coefficients to the p-weighted terms.

- If a physical relationship is constant (e.g., conservation of mass), the coefficient for the p-term will be zero.
- If a relationship degrades (e.g., Fuel Flow vs. Rotor Speed), the algorithm will assign a non-zero coefficient to the $p \cdot W_f$ term, effectively learning: $\dot{N}_f = \alpha W_f + \beta (p \cdot W_f)$.
  This allows the model to seamlessly interpolate between healthy and degraded physics.

5.3 Analytic Redundancy and Fault Isolation

This Adaptive Monitor enables robust **Analytic Redundancy** to distinguish sensor faults from mechanical failures.

The Logic of Disagreement:

1. **Scenario: True Engine Degradation.**
   - The engine wears. EGT rises. The calculated $p(t)$ decreases.
   - The LPV model sees the new $p(t)$ and adjusts its expected dynamics using the Ξ_1 terms.
   - The predicted sensor values match the measured values. **Residuals are Low.**
   - **Conclusion:** The engine is degrading. The Mamba RUL prediction is valid.
2. **Scenario: Sensor Drift (False Positive).**
   - The T_{50} sensor drifts high due to a fault, but the engine is healthy.
   - The calculated $p(t)$ decreases artificially.
   - The LPV model, seeing a "low" $p(t)$, expects the *other* sensors (like Rotor Speed N_f and Compressor Pressure P_{30}) to behave like a dying engine (e.g., requiring higher fuel flow).
   - However, the *actual* N_f and P_{30} sensors are reading nominal values (because the engine is actually healthy).
   - The model predicts "Dying Behavior"; the sensors show "Healthy Behavior." **Residuals are High.**
   - **Conclusion:** The physics are inconsistent. This is a sensor fault, not engine failure. The Decision Module masks the T_{50} sensor and alerts maintenance, avoiding an unnecessary flight abort.

---

6. Software Architecture: The Simulation Fabric

The robustness of the SIL environment is just as critical as the accuracy of the algorithms. SAFER v3.0 abandons the brittle "compile-and-link" approach (DLLs) in favor of a modern, multi-process architecture.

6.1 Process Isolation and Stability

The system runs as two independent OS processes.[1]

- **Process A (Plant):** The C-MAPSS Simulation.
- **Process B (Guardian):** The SAFER v3.0 Algorithm Stack (Mamba + SINDy).

Why Separation?

1. **Fault Tolerance:** In a research environment, experimental neural networks can crash (CUDA out-of-memory, segmentation faults). If this happens in a monolithic application, the simulation crashes, and the test data is lost. In a split architecture, if the Guardian crashes, the Plant continues to log data, allowing post-mortem analysis of *what* engine state caused the crash.

2. **The GIL:** Python's Global Interpreter Lock limits threads to one CPU core. Separate processes allow the Simulation and the Neural Network to run on separate physical cores, maximizing throughput.
3. **Startup Safety:** We enforce the spawn start method (mp.get_context('spawn')). The default fork on Linux copies the parent's memory, which can corrupt the internal state of complex libraries like PyTorch or OpenMP (leading to deadlocks). Spawn creates a fresh interpreter, ensuring clean initialization of the BLAS libraries.

6.2 The Shared Memory Transport

Data transfer is handled via multiprocessing.shared_memory, which maps a block of RAM into the address space of both processes. This is **Zero-Copy**. Unlike Sockets or Pipes, data is not serialized (pickled) or copied into kernel space. It is written directly to RAM by the Plant and read directly by the Guardian. This reduces latency from milliseconds to microseconds.

6.3 The Lock-Free Ring Buffer

To handle the potential timing jitter between the hard-real-time Plant and the soft-real-time Guardian, we implement a **Circular Buffer (Ring Buffer)** in the shared block.

Byte-Level Memory Map:
The shared memory block is a raw byte array structured as follows:

Offset (Hex)	**Field Name**	**Data Type**	**Description**
0x0000	Write_Head	int64 (8 bytes)	Atomic counter. Monotonically increasing index of the next write slot.
0x0008	Read_Tail	int64 (8 bytes)	Atomic counter. Monotonically increasing index of the last read slot.
0x0010	Buffer_Size	int64 (8 bytes)	Total number of frames the buffer can hold (e.g., 1024).
0x0018	Flags	uint8	Status flags (Bit 0: Overflow, Bit 1: Reset).

0x0020	Reserved	bytes	Padding for 64-byte cache line alignment.
0x0100	Frame_0	float32	Sensor Data Vector for $t=0$ (56 bytes).
0x0138	Frame_1	float32	Sensor Data Vector for $t=1$.
...	...	...	...

The Protocol:

1. **Producer (Plant):**
   - Calculates write index: idx = Write_Head % Buffer_Size.
   - Writes sensor data to Frame[idx].
   - **Memory Barrier:** Ensures data write completes before updating head.
   - Increments Write_Head. (Atomic operation).
2. **Consumer (Guardian):**
   - Reads Write_Head.
   - Checks if Write_Head > Read_Tail.
   - If true, calculates idx = Read_Tail % Buffer_Size.
   - Reads Frame[idx].
   - Increments Read_Tail.

This protocol is **Lock-Free**. The Producer *never* waits for the Consumer. If the Consumer hangs, the Producer continues writing. If the buffer fills (Write_Head - Read_Tail >= Buffer_Size), the Flags register is set to Overflow, alerting the operator, but the simulation does not crash. This mimics the behavior of a real flight data recorder (circular overwrite).

---

7. Runtime Assurance: The Simplex Decision Module

The "Simplex Architecture" is a proven safety pattern for autonomous systems. It wraps a high-performance "Complex" controller (Mamba) with a formally verifiable "Simple" safety core.

7.1 The Safety Baseline

The "Simple" core is a Physics Trend Extrapolator. It uses the EGT Margin gradient.

$$RUL_{Base} = \frac{EGTM_{Current}}{\max(\epsilon, |\frac{d}{dt}EGTM_{Smoothed}|)}$$

While this model is less accurate (it assumes linear degradation, whereas real degradation accelerates), it is monotonically consistent. It effectively sets a "Speed Limit" on the RUL. If Mamba predicts RUL=500 but the EGT is dropping like a stone, the Baseline will predict RUL=50. The Decision Module will detect this discrepancy.

7.2 Uncertainty Quantification: Adaptive Conformal Prediction

A raw number ("RUL = 124") is dangerous without confidence bounds. SAFER v3.0 uses Adaptive Conformal Prediction (ACP) to generate a rigorous prediction interval.

$$C(x_t) = [\hat{y}_t - \delta_t, \hat{y}_t + \delta_t]$$

The width δ_t is updated online based on the recent performance of the model.

$$\delta_{t+1} = \delta_t + \lambda (\alpha - \mathbb{I}\{y_t \in C(x_t)\})$$

- If the true value falls *outside* the interval, δ increases (widens).
- If the true value falls *inside*, δ decreases (tightens).
- λ: Step size (learning rate).
- α: Target error rate (e.g., 0.05 for 95% confidence).

This provides a dynamic "Trust Metric." If the engine enters a novel operating regime that Mamba hasn't seen, the error rate will spike, the interval will automatically widen to cover the uncertainty, and the pilot will see a "Low Confidence" indicator.

7.3 Alert Prioritization Matrix

The inputs from Mamba, LPV-SINDy, and the Baseline are fused into actionable alerts.

Priority	Alert Name	Condition	Logic	Action
P1	**CRITICAL FAILURE**	$RUL_{Mamba} < 20$ **AND** $EGTM \approx 0$	Models Agree. Physics confirm imminent failure.	**Return to Base (RTB).**

P2	**DEGRADATION WARN**	$RUL_{Mamba} < 50$ **AND** LPV Ξ_1 Active	Early warning. Physics monitor detects parameter shift.	**Schedule Maintenance.**
P3	**MODEL CONFLICT**	$	RUL_{Mamba} - RUL_{Base}	> \tau$
P4	**SENSOR FAULT**	LPV Residuals High (Single Channel)	Physics violation localized to one sensor.	**Mask Sensor.** Continue operation using redundant sensors.

8. Implementation Strategy and Verification

8.1 Implementation Roadmap

The deployment of SAFER v3.0 follows a rigorous V-model approach.

1. **Phase 1: Fabric Verification.** Build the Python Multiprocessing harness. Validate the Ring Buffer at 1000Hz. Ensure zero data corruption over a 24-hour stress test.
2. **Phase 2: Core Training.** Train the Mamba model on C-MAPSS FD004 (the most complex subset with 4 operating conditions). Use the Parallel Scan implementation. Target RMSE < 15 cycles.
3. **Phase 3: Monitor Calibration.** Run LPV-SINDy on the training data. Extract the Ξ_0 and Ξ_1 matrices. Tune the residual thresholds for fault isolation using synthetic fault injection (adding bias to test data).
4. **Phase 4: SIL Integration.** Connect Plant and Guardian. Execute full run-to-failure simulations. Verify that the Decision Module correctly switches priorities (e.g., masking a drifting sensor).

8.2 Validation Metrics

- **Prognostic Accuracy:** RMSE and NASA Scoring Function.
- **Computational Latency:** Real-Time Factor (RTF). The total time for (Read Buffer + Mamba Inference + SINDy Check + Decision Logic) must be $< 20ms$ on a standard Intel i7 laptop CPU.
- **Safety Coverage:** Percentage of injected sensor faults correctly identified by the LPV

monitor (Target > 99%).

---

9. Conclusion

SAFER v3.0 represents the definitive maturation of data-driven prognostics. It moves beyond the naive application of "Black Box" neural networks, addressing the specific physical and computational constraints of the aerospace domain.

- By replacing Transformers with **Mamba**, it solves the sequence length bottleneck and ensures deterministic inference latency.
- By replacing Static SINDy with **LPV-SINDy**, it solves the "Healthy vs. Dying" observability conflict, enabling robust physics-based monitoring.
- By replacing compiled DLLs with **Shared Memory Multiprocessing**, it ensures a robust, cross-platform SIL environment.

This architecture is not merely a research proposal; it is a finalized, ready-to-implement specification that bridges the gap between the stochastic promise of AI and the deterministic requirements of flight safety.

End of Report.

Works cited

1. SAFER v3.0 SIL Architecture Design.pdf