Confidence Audit: Uncertainty Propagation and Feedback Architecture

Where in the pipeline confidence can diverge from accuracy, and what closes the gap

Premise

The digital twin described in the accompanying architectural report was specified for a TRAK VMC7si machining aluminum. It was trained on a mosaic of proxy datasets: CMAPSS turbofan degradation for the primary RUL demonstration, NUAA Ideahouse titanium milling for CNC-specific RUL, TUAWS aluminum cutting for product classification, and UC Berkeley milling data for wear multipliers. Each component is validated individually against its training domain. None has been validated against the target cell.

This is expected. The frozen-backbone architecture is designed so that domain heads can be swapped without touching the shared encoder, and the four fidelity levels are sequenced so that no capability deploys before the data supporting it is in place. But it means that at deployment, the system carries structural uncertainty that cannot be resolved by architecture alone. The wear multipliers are borrowed from a different machine. The classifier has never seen this cell's sensor signatures. The RUL model's training distribution does not include this specific tooling, fixture, and coolant configuration.

These are the boundary conditions that define what the feedback architecture must resolve. This document traces where in the pipeline that structural uncertainty lives, how it interacts across layers, and what mechanisms narrow it over operational time.

The argument in brief:

Problem	Response
Uncertainty changes character at model boundaries. Sensor noise is additive; at the classifier it becomes a routing decision (wrong trajectory by ×2.4); at the multiplier layer it embeds as an invisible assumption. Per-model confidence checks pass while the aggregate prediction drifts.	Prediction error at each layer becomes a first-class input to current confidence. Every tool replacement, classifier verification, and operator override measures the gap between prediction and outcome. Assumed confidence from training data is replaced by measured confidence from this cell, this tooling, these products. Compound error becomes visible because the output is checked.
The wear multipliers are borrowed constants from a different machine and material, embedded in inference logic where no confidence check can see them. Every internal metric looks stable while every output carries a proportional bias.	The multipliers get uncertainty bounds and an empirical replacement path. Sensitivity analysis makes the assumptions visible; product-segmented replacement data accumulated over months replaces the proxy constants with values calibrated to this specific cell.
The scheduler consumes RUL point estimates and discards the confidence interval. If the tool fails five cycles early, the system produces exactly the unplanned downtime it was designed to prevent.	The hard constraint becomes "maintenance completes before the lower bound of the confidence interval," parameterized by the cost asymmetry between unplanned failure and early replacement, adjustable by operational context and operator risk tolerance.
Human overrides create censored data. Operators who replace tools early prevent the system from observing true end-of-life behavior, biasing training data toward conservative estimates, which reinforces the distrust that caused early replacement.	Survival analysis replaces naive averaging. Kaplan-Meier and Cox proportional hazards models handle right-censored observations natively, so early replacements and run-to-failure events both update the tool life curve.

Uncertainty Propagation

The value proposition is specific: convert unplanned maintenance into scheduled maintenance in a way that optimizes job and order completion. That proposition holds only if the RUL predictions are accurate enough, and known to be accurate enough, that a maintenance schedule built on them is trustworthy. The question is where in the model pipeline uncertainty can accumulate to the point where it breaks that trust.

Sensor Layer: Additive Noise

Measurement noise, calibration error, drift, and staleness. The sensor integrity module handles this through cross-sensor consistency checks, staleness tracking with configurable freshness windows, and the four-state epistemic flag. The architecture catches uncorrelated drift: when one sensor diverges while others agree, the system isolates the drifted source before its error propagates downstream.

The harder case is correlated drift, where all sensors shift together. A thermal event affecting the entire sensor suite, a shared power rail introducing common-mode noise, gradual fixture loosening that changes the mechanical transfer path from cutting zone to accelerometer: these produce coordinated shifts that cross-validation cannot catch because there is no disagreeing signal. The system's response in this case is honest but limited. It can widen confidence intervals and escalate to human judgment. It cannot quantify how wrong it might be, because the reference against which wrongness would be measured is the one that moved.

Product Classifier: The Routing Switch

This is the most consequential uncertainty in the pipeline due to the character and consequences of error.

Every other model in the pipeline produces uncertainty that is additive: the RUL estimate is off by some amount, the envelope monitor's threshold has some tolerance, the sensor health check has a detection delay. These uncertainties widen the confidence band around a prediction, but the prediction remains in the right neighborhood. A classifier error is structurally different. Rather than adding noise to the RUL estimate, it selects a different wear trajectory entirely. If the system classifies Product A (multiplier 1.0×) when Product C is actually running (multiplier 2.406×), the RUL estimate is wrong by a factor of 2.4, and it remains wrong for the entire duration of the misclassification.

At 99.93% validation accuracy, this seems safe. But that accuracy was measured on TUAWS training data under controlled conditions. The real question is what happens at the boundaries: during changeover periods where the sensor signature does not cleanly match any product type, during unusual blank lots that shift the vibration profile, or as the classifier's training distribution ages and the machine's sensor signatures drift. The architecture handles low-confidence classification correctly, gating below-threshold predictions and escalating to operator verification. What it cannot catch is a confident misclassification: a case where the classifier assigns the wrong product with high probability because the sensor signature genuinely resembles a different product under the current conditions.

This is a specific instance of the confidence-consequence inversion pattern. The system is most dangerous not when it is uncertain (uncertainty triggers the right safeguards) but when it is certain and wrong (certainty bypasses them).

RUL Model: Epistemic and Aleatoric Uncertainty

The TwinnableTFT and Stack-LSTM produce point estimates with associated confidence via MC dropout. Two kinds of uncertainty are present. Aleatoric uncertainty, the irreducible process variability from blank lots, coolant degradation, and ambient temperature, appears as variance around the prediction that no model can eliminate. Epistemic uncertainty, from limited training data or out-of-distribution inputs, appears as high MC dropout variance: the model produces different answers on different forward passes because it lacks the evidence to commit.

The epistemic layer captures both. Mahalanobis distance checks whether the current sensor window is statistically anomalous relative to the training distribution. MC dropout quantifies how stable the model's output is under perturbation. Together they distinguish between a confident prediction on familiar data (low distance, low variance: trust the output) and an unstable prediction on unfamiliar data (high distance, high variance: widen the band and flag for review).

What neither mechanism can detect is a systematic bias that is consistent across forward passes. If the model was trained on turbofan degradation curves and deployed on aluminum milling, the predictions may be precise (low MC dropout variance) and in-distribution by the model's own assessment (low Mahalanobis distance relative to its training set), yet systematically offset from the physical process they claim to describe. The model does not know that its training domain and its deployment domain are different. It only knows that the inputs look normal by its own standards.

Wear Multipliers: The Invisible Assumption

The UC Berkeley-derived constants (1.0, 2.189, 2.406) are treated as known values in the current architecture. They are not. They are estimates derived from a different machine, different tooling, and potentially different aluminum alloys. The true multipliers for the TRAK VMC7si with its specific tooling and cutting programs might be 1.0, 1.8, and 3.1.

This uncertainty is uniquely dangerous because of where it sits in the pipeline. It is an assumption embedded in the inference logic itself, invisible to every confidence check the system performs. The Mahalanobis distance does not flag it because the inputs look normal. MC dropout does not flag it because the model gives consistent outputs. The sensor integrity layer does not flag it because the sensors are healthy. Every internal metric looks stable while every output carries a bias proportional to the gap between the assumed multiplier and the true one. As noted in the full report- this known assumption is addressed as part of the initial calibration phase of deployment.

Scheduler: Point Estimates and Hard Constraints

The OR-Tools CP-SAT solver treats the RUL prediction as a hard constraint: maintenance must complete before RUL reaches zero. The RUL prediction is a distribution rather than a point value. The scheduler currently consumes the point estimate and discards the confidence interval.

The consequence is direct. If the model says RUL = 50 cycles with ±10% accuracy, the scheduler plans for 50. If the tool fails at cycle 45, the maintenance window arrives five cycles too late, and the system has produced exactly the unplanned downtime it was designed to prevent. If the scheduler always used the conservative bound (RUL = 45), it would avoid surprises but replace tools with 10% of their life remaining, eroding the efficiency gains that justify the system's existence.

The optimal behavior sits between these extremes, and it depends on a quantity the system does not currently model: the cost asymmetry between unplanned failure and early replacement. A tool failure during a critical order with a tight delivery window is categorically worse than the same failure during a low-priority run with schedule slack. The risk tolerance should be parameterized by operational context instead of being fixed as a system constant.

Two Categories of Uncertainty

The pipeline's uncertainties divide into two categories that require different architectural responses:

Observable (epistemic layer catches)	Structural (passes through undetected)
Single-sensor drift: cross-sensor consistency checks isolate the drifted source	Correlated drift: all sensors shift together, producing no disagreeing signal
Low classifier confidence: flagged and escalated to operator verification	Confident misclassification: wrong wear trajectory selected with high probability (multiplicative ×2.4 bias)
High MC dropout variance: model uncertainty quantified and surfaced	Domain gap: training distribution differs from deployment, but model's internal metrics look stable
	Multiplier calibration bias: fixed constants from a different machine embedded in inference logic

Compound Error

The structural uncertainties interact. A marginal misclassification (the classifier assigns Product B at 85% confidence when the true product is C) combined with early-stage sensor drift (not yet flagged by the integrity layer) combined with a wear multiplier that is 15% miscalibrated for this specific machine produces an RUL estimate that passes every individual model checkpoint. No single model is visibly failing. The Mahalanobis distance is within bounds. The MC dropout variance is low. The sensor integrity panel shows green. The compound error is large enough to invalidate the maintenance schedule, and the system's architecture provides no mechanism to detect it.

The system's confidence indicators are evaluated per-model, rather than across the pipeline. Therein lies the gap. Each layer evaluates its own uncertainty without knowledge of how its output interacts with the uncertainties of the layers above and below it.

Three Interventions

Consume the confidence interval, not the point estimate. The scheduler should consume the RUL confidence interval rather than the point estimate. The hard constraint becomes "maintenance completes before the lower bound of the confidence interval," parameterized by an operator-adjustable risk tolerance that reflects the cost asymmetry between unplanned failure and early replacement.

Bound the wear multipliers. The wear multipliers should carry uncertainty bounds. Even a first-order sensitivity analysis ("if the true Product C multiplier is 2.0 instead of 2.4, the maintenance window shifts by X cycles") makes the system's assumptions visible and auditable. The multipliers are currently invisible to the confidence architecture; bounding them brings them into the epistemic layer's scope.

Accumulate prediction error as a first-class confidence input. Prediction error at each layer should feed into current confidence. Every tool replacement measures the gap between predicted and actual RUL. Every classifier verification measures real-world accuracy. Every operator override reveals where the model's state-to-outcome mapping was wrong. These signals, accumulated over months, replace assumed confidence with measured confidence for this specific cell, this specific tooling, these specific products.

Feedback Architecture

The infrastructure for feedback is already in place: the maintenance log records operator decisions with the RUL at time of decision, the digital thread preserves the causal chain from sensor reading to prediction to action, and the tool magazine tracks per-tool run history across replacements. What remains is wiring the outcomes of those decisions back into the models that informed them.

Four feedback channels connect operational ground truth to specific pipeline layers, each accumulating at a characteristic rate.

Channel	Signal	Target	Rate	Converges
Confirmed labels	Operator verifies product at changeover	Product classifier retraining	Per shift	Weeks
Prediction error	Predicted vs actual RUL at replacement	RUL model bias correction	Per replacement	Weeks to months
Measured confidence	Accumulated prediction error distribution	Epistemic gate calibration	Accumulated	Months
Risk profile	Operator accept/defer/override with RUL at time of decision	Scheduler risk tolerance	Per override	Weeks

Classifier Feedback: The Fastest Loop

Every changeover where an operator confirms or corrects the product classification is a labeled training sample. The accumulation rate is fast: potentially dozens of verified classifications per shift. Within weeks of deployment, the classifier's real-world accuracy is empirically measured rather than estimated from training data.

The system should track misclassification rate as a rolling metric segmented by tool life stage (the classifier might degrade with worn tools, since the sensor signature it trained on was dominated by fresh-tool conditions), by transition period (the first seconds of a new product where the signature is ambiguous), and by time since last retraining. If the rolling rate climbs above a configurable threshold, a retraining flag fires. This is the first structural uncertainty that can be empirically narrowed, because the data arrives fastest.

RUL Prediction Error: The Core Calibration Signal

Each tool replacement produces a prediction-error measurement: the gap between what the model predicted at time T and the tool's actual remaining life at time T. One data point per replacement, arriving over weeks to months. After 20 to 50 replacement events, the error distribution stabilizes enough to reveal whether errors are systematic (the model consistently overestimates RUL by 12%, suggesting the multipliers are too conservative or the model carries a domain-transfer bias) or high-variance but unbiased (proper calibration with inherent process uncertainty). The former is correctable through bias adjustment. The latter defines the irreducible floor of the system's prediction accuracy.

Segmenting this error by product mix, tool type, and sensor health state reveals which conditions produce the largest errors. If the model is accurate on single-product runs and inaccurate on mixed-product sequences, the composite wear aggregation logic is the problem, not the base wear curves. If accuracy degrades as sensor confidence drops, the epistemic layer's gating thresholds may be too permissive. The prediction error signal diagnoses the pipeline.

Measured Confidence: From Assumption to Empiricism

As prediction errors accumulate, assumed confidence can be replaced by measured confidence. The system transitions from "I think I am ±10% accurate because that is what the training data suggested" to "I am ±8% accurate on Product A and ±14% on Product C, based on the last 40 runs of each." This measured confidence is what should parameterize the scheduler's risk tolerance. It is also what converts the structural uncertainty column from the propagation analysis into bounded, empirically calibrated quantities.

The convergence rate is set by the slowest-accumulating product type. If Product C runs infrequently, the system's confidence in the Product C multiplier and wear curve will lag behind the others. The architecture should surface this asymmetry rather than reporting a single aggregate confidence: the system's accuracy is not uniform across products, therefore the scheduler's risk tolerance should not be either.

Human Overrides as Prediction-Error Signals

An override where a technician keeps a tool running past the model's recommended replacement is the most valuable training event the system can receive. The tool's subsequent behavior directly measures the model's error at the decision point.

Three outcomes are possible, and each teaches something different. The tool survives well past the predicted failure point: the model was conservative for this product-tool-condition configuration, and the prediction carried a systematic bias. The tool fails shortly after the predicted point: the model was approximately correct, and the technician accepted a risk that happened not to materialize. The tool fails before the predicted point: both the model's prediction and the technician's override compounded the error.

The inverse override, where a technician replaces a tool that the model rated as healthy, is equally informative but about a different layer. If post-replacement inspection reveals damage the sensors did not detect, the technician has identified a sensor coverage gap. The system should log these events separately and flag them as sensor augmentation candidates.

The Censored Data Problem

Human overrides create a censored data problem. A tool replaced at 80% of its predicted remaining life is a right-censored observation: the system knows the tool survived to that point but not when it would have actually failed. If operators consistently replace tools early because they do not trust the model, the system never observes true end-of-life behavior. Its training data becomes biased toward conservative replacements, which makes the model more conservative, which reinforces the operators' distrust. The loop is self-perpetuating.

Survival analysis methods handle this natively. Kaplan-Meier estimators and Cox proportional hazards models are designed for datasets where some observations are complete (tool ran to failure) and others are censored (tool replaced before failure). Maintaining a survival curve per tool type and product configuration allows every replacement event to update the curve regardless of whether the replacement followed the model's recommendation, preceded it, or contradicted it.

Operator Risk Tolerance

The maintenance log reveals a quantity the system cannot get from sensors: each operator's implicit risk tolerance. The distribution of RUL values at which a specific operator accepts versus defers maintenance builds a behavioral profile. These profiles are context-dependent. The same operator may defer freely during low-priority runs and accept maintenance conservatively when a critical order is active. Each operator’s risk profile is a function of operational context and learned expertise.

The system should model this tolerance rather than adopting it. An operator who consistently defers to RUL = 12 and whose deferred tools survive is surfacing information about the model's conservatism. An operator whose deferrals correlate with increased tool failures is exceeding the physical envelope. The system's role is to make the cost of each decision visible: "Deferring maintenance here saves 45 minutes of downtime but increases tool failure probability from 3% to 12%, based on the last 30 similar deferrals." The operator decides. The system ensures the decision is informed.

Trust Calibration

As predictions improve through feedback, operators trust the model more, generating fewer overrides, which means less override data, which means slower learning from overrides. This is exploration-exploitation in disguise, and the architecture handles it naturally.

The epistemic gate communicates confidence. When confident, operators follow recommendations, which is appropriate because the model is probably right. When uncertain, operators apply their own judgment, which generates the most informative training data: concentrated at the frontier where the model is genuinely uncertain and additional evidence has the highest value. The feedback rate is self-regulating. Fast early learning, when everything is uncertain and overrides are frequent, tapering to targeted learning at the model's genuine uncertainty frontier.

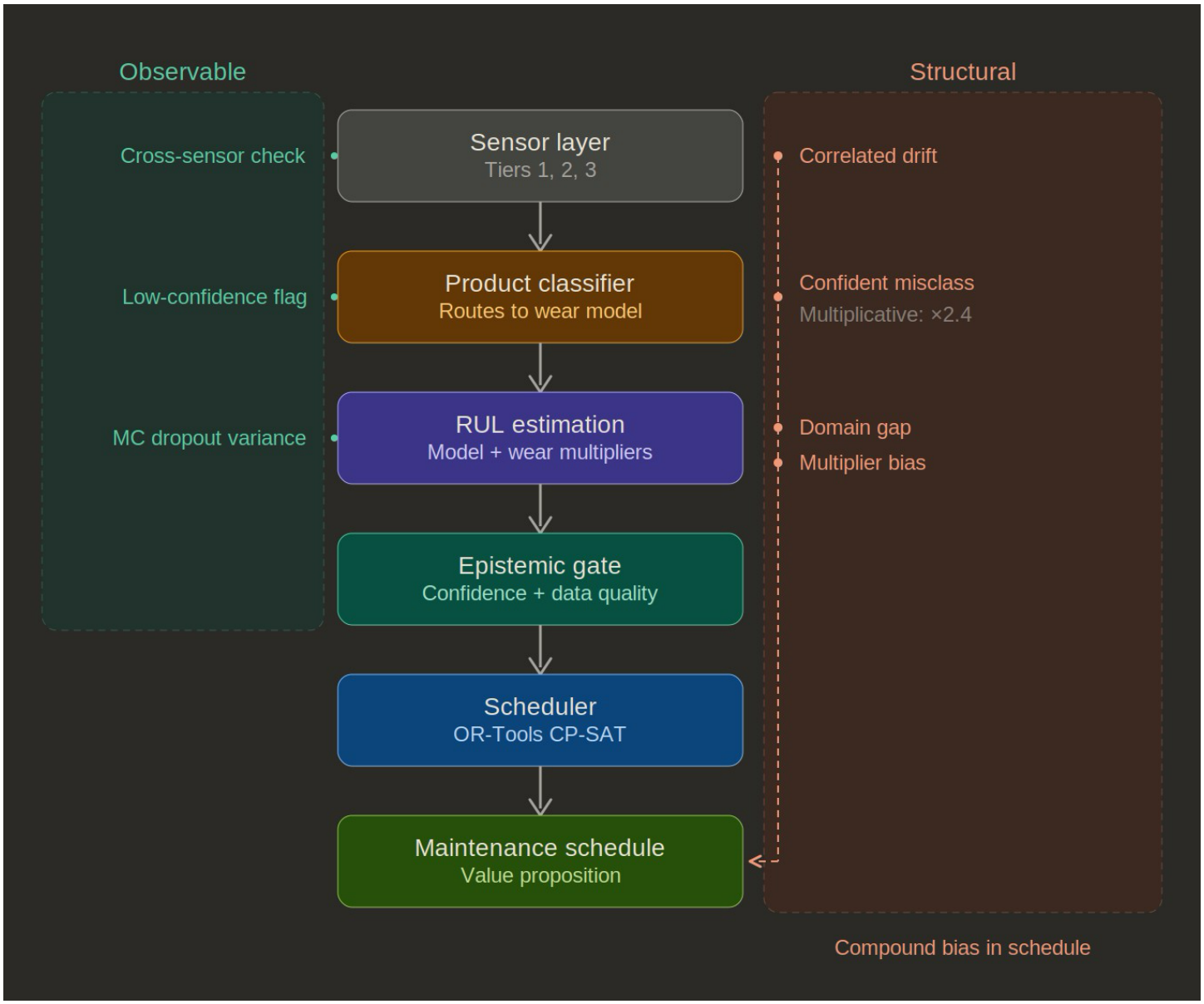
The Flywheel

The feedback channels do not improve their own layer in isolation. Better classifier accuracy means the correct multiplier gets applied, which means more accurate RUL, which means better-timed maintenance. Better RUL accuracy means tool replacements at more informative points in the tool's life cycle, which means higher-quality ground truth for multiplier recalibration. Better multiplier calibration means more accurate RUL in mixed-product runs, which means fewer overrides driven by operator intuition correcting for systematic model bias.

The structural uncertainty column from the propagation analysis narrows over time, because each improvement at one layer propagates through the pipeline and reduces compound error everywhere downstream. No single feedback mechanism is transformative. The compounding is. The rate of convergence is set by the slowest channel: multiplier recalibration, which needs product-segmented replacement data accumulated across months of operation. The system gets more honest about its own accuracy the longer it runs, and its honesty compounds.

Diagrams of Pipeline Uncertainty

Uncertainty Propagation



Uncertainty Reduction Feedback Loops