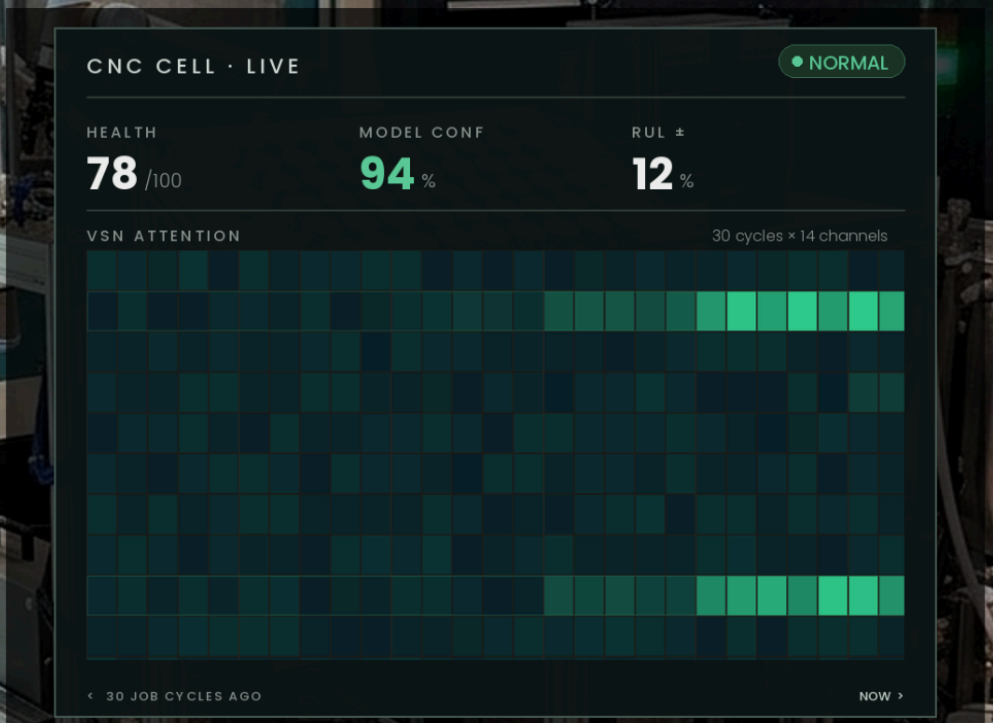
Digital Twin System for CNC Manufacturing Cell



Predictive Maintenance and Queue Optimization Through Sensor-Driven Digital Twin Architecture

Andrew Clark · Project Lead
Dan Brown · Lead Software Engineer
Rodrigo Perez · Domain Expert

Spring 2026 · CMU 12-830A · Principles of Digital Twins

Section 1: Project Definition and System Context

Part 1.1: Problem Definition and System Context

The Cell and Its Stakeholders

In its current state, the CNC manufacturing cell described in this report has its cutting tools replaced on a calendar schedule, has its jobs sequenced by planner intuition, and cannot tell when its own sensors have drifted from the physical reality they claim to report, resulting in emergency maintenance and repair being conducted reactively. The digital twin proposed here closes each of these gaps.The cell is a closed world: a self-contained cluster of machines, robots, and material handling equipment organized to produce finished parts from raw stock in a single automated workflow. Blanks enter from a hopper. Two collaborative robots (cobots) from Universal Robots load them into a high-performance vertical CNC milling machine equipped with a Human Machine Interface (HMI) and Automatic Tool Changer (ATC). Inside the machine, appropriate tools are loaded, the spindle cuts, the arms extract the finished part, clean it, and deposit it in an outbox. This particular cell manufactures three distinct products from similar-size aluminum blanks, each requiring different tooling, tool paths, and potentially different alloy grades. At any given moment, one of three cutting programs is running, consuming one of three sets of tools at one of three wear rates, producing one of three products against one of three delivery schedules.

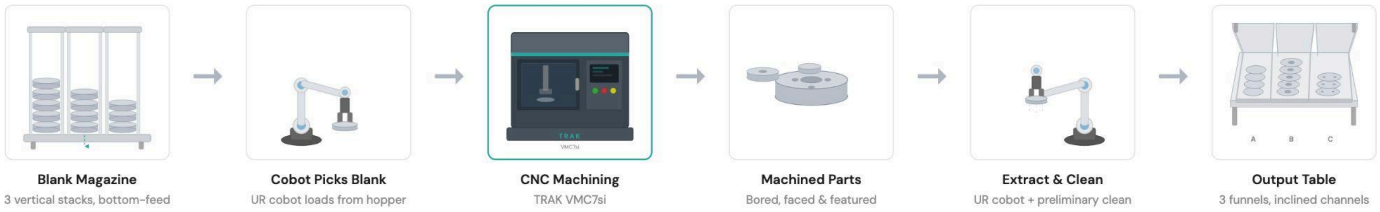


Figure 1.1. Physical concept of the digital twin system

The cell’s operators, maintenance technicians, production planners, and operations managers all depend on this same physical system, but they see it at different scales. An operator watches the current cycle: vibration, sound, chip formation, the sensory reality of metal being removed. A maintenance technician tracks tool condition across shifts, looking for the slow drift that precedes failure. A production planner manages the order queue across days, balancing due dates against changeover costs. Operations management evaluates equipment effectiveness across quarters, translating uptime and scrap rates into financial performance. Each stakeholder makes decisions at a different timescale, and the information each needs flows from the same sensor data through different analytic lenses.

The digital twin proposed here is the system that performs those translations: from raw sensor observation to tool life estimate, from tool life estimate to maintenance recommendation, from maintenance constraint to schedule optimization. It does not replace any stakeholder’s judgment. It makes that judgment possible at a speed, resolution, and duration that manual monitoring cannot sustain.

The system must also specify the return path: the channels through which human actions flow back into the model's state. When a technician confirms a tool replacement, the remaining useful life (RUL) counter resets. When an operator dismisses a false alarm, that dismissal adjusts future alert thresholds. When a job completes late, the scheduler’s timing estimates update. When a machine status override contradicts the

model’s inferred state, the discrepancy feeds back into the classifier. When an operator logs shift context (a new blank lot, a fixture change, an unusual ambient condition), the twin acquires information that sensors alone cannot provide. These five operator input types (maintenance confirmation, anomaly acknowledgement, job completion report, machine status override, and shift context) must each be defined with UI placement, data schema, and downstream effect. Without this return path, the twin cannot improve from operational experience.

Why Current Methods Fall Short

Without a digital twin, the cell’s maintenance and scheduling operate on separate, uncoordinated logics. Tool replacement follows either a calendar interval or a run-hour threshold: replace every N hours of spindle time regardless of what was being cut and how it was being cut. This approach is simple and conservative. It is also structurally unable to account for the fact that an hour of finish cutting Product A in aluminum does not consume tool life at the same rate as an hour of roughing and profiling Product C. A tool that has machined thirty minutes of one product and twenty minutes of another carries a composite wear state that neither product’s individual wear curve describes. Calendar-based replacement either changes tools with significant remaining life, wasting tooling cost and productive time, or misses accelerated degradation from a demanding product sequence, risking mid-cycle failure and scrap.

The job queue faces a parallel limitation. Currently, sequencing decisions rely on due dates, order priority, and planner experience. What the planner cannot easily compute is the downstream effect of sequence on tool consumption. Running similar products consecutively reduces changeover time but may delay a higher-priority order. Interleaving products balances delivery risk but may accelerate tool wear through frequent transitions between cutting profiles. The planner optimizes for delivery. The maintenance schedule optimizes for tool life. Neither optimization accounts for the other.

What the System Cannot Know About Itself

The deeper limitation is one of self-knowledge. The cell has no mechanism to detect when its own sensors have drifted from the physical reality they claim to report, and this is the problem the digital twin's architecture must address most carefully. The problem is epistemological, and it is shared by every system that acts on measurements it cannot independently check.

A vibration sensor subject to mounting degradation does not report “unreliable.” It reports a shifted value that a downstream model would interpret as a changed cutting condition. A performance envelope calibrated against fresh tooling becomes misleading as tools wear, because the baseline has shifted but the envelope has not.

These are not failures of data collection. They are failures of self-assessment: the system’s inability to distinguish between a world that has changed and a sensor that has drifted.

This distinction matters because the two failure modes demand opposite responses. A system that knows its sensor is degraded can widen margins, flag uncertainty, or escalate to human judgment. It is diminished but honest. A system that trusts a drifted sensor is dangerous in proportion to its confidence: its internal metrics look stable, its model looks consistent, its recommendations look reasonable, and every one of them is anchored to a physical reference that no longer exists.

Sensor drift is not the only form of silent divergence. The predictive model itself can drift: its learned parameters, calibrated against one distribution of cutting conditions, may become progressively less representative as the cell's product mix, tooling, or material supply evolves. Sensor drift and model drift are

distinct failure modes requiring distinct detection mechanisms, but both produce the same architectural consequence: a system whose confidence no longer tracks its accuracy.

The architecture of the digital twin must ensure that epistemic failure, the state where the system is confidently wrong, is architecturally prevented or, where it cannot be prevented, made detectable. These are two distinct engineering problems.

Prevention means designing redundancy into the sensor layer so that no single feed can silently corrupt the model. Cross-validation between independent sensor streams (vibration, power draw, RPM) provides this: during a known cutting condition, these feeds should correlate in predictable ways. When one feed diverges while the others agree, the architecture can isolate the drifted source before its error propagates downstream. Prevention also means indexing performance envelopes to tool life stage rather than calibrating them once against fresh tooling. An envelope that adapts to expected wear progression does not mistake normal degradation for anomaly, and does not mistake sensor drift for normal degradation.

Detection applies where prevention is not possible: where all available feeds drift together (a thermal event affecting the entire sensor suite), where the physical system changes in ways no sensor is positioned to observe (gradual fixture loosening, coolant degradation), or where the model's own assumptions have become stale. Detection requires the system to maintain an explicit model of its own confidence, updated at every inference step, and to surface that confidence to operators as a first-class output rather than burying it in metadata. The architectural concept sketch (see Figure 1.2) illustrates this through the model confidence layer, which gates every prediction leaving the digital twin engine with both a value and a data-quality score.

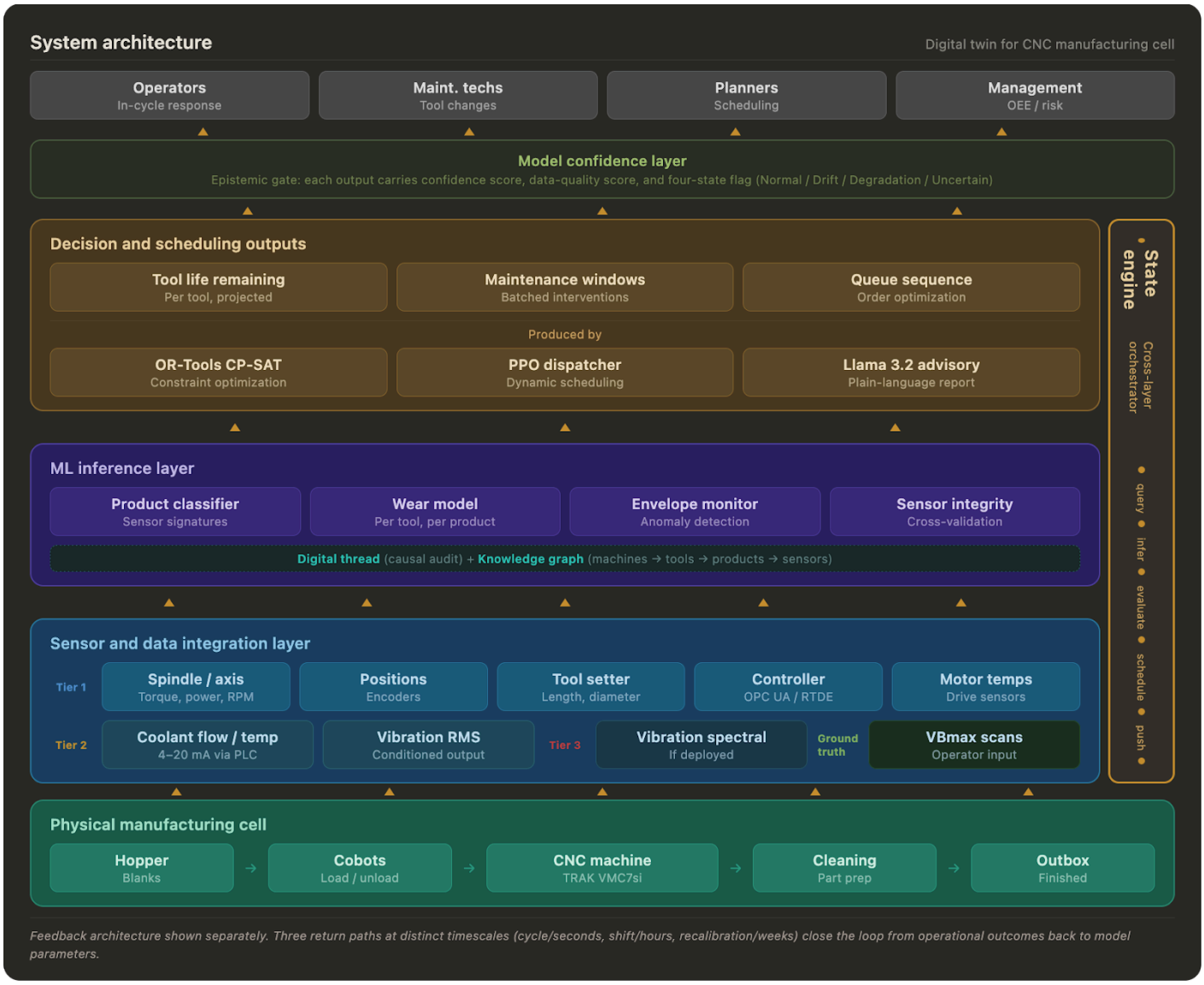


Figure 1.2. Digital twin system architecture. Four functional layers from physical cell through stakeholder interface, with the model confidence layer operating as a cross-cutting gate. The model confidence layer gates every prediction with a data-quality score. The feedback loop represents operator and maintenance actions that alter the physical system the twin models.

Deployment Roadmap: Four Fidelity Levels

The digital twin's capabilities map to four named fidelity levels, each with distinct data requirements, model dependencies, and acceptance criteria. These levels are delivered in sequence because each depends on the one before it.

Behavioral. The twin observes and displays. It connects existing sensors and the CNC controller into a unified view, providing real-time visibility into cell state, sensor readings, and job status. This is the descriptive twin: it adds value immediately through connectivity and data access, before any predictive model exists. Every capability that follows is an added layer on this foundation.

Predictive. Sensor data flows into models that estimate remaining tool life, classify active product type, and monitor performance envelopes. The twin tells stakeholders what is likely to happen next.

Prescriptive. RUL estimates drive maintenance scheduling recommendations. The twin tells stakeholders what they should do and when.

Optimizing. Maintenance constraints, tool life projections, and order priorities feed a queue optimizer that sequences jobs to maximize full order completions within delivery windows. The twin acts within defined bounds, requesting human approval only when its confidence drops below threshold.

Each level has a deployable deliverable. The behavioral twin is useful from day one. The predictive twin requires labeled training data and calibrated wear models. The prescriptive twin requires validated RUL accuracy. The optimizing twin requires defined priority weights and stakeholder agreement on what “optimal” means. This sequencing ensures that no capability is deployed before the data and validation supporting it are in place.

Decision Focus

The digital twin supports three interconnected classes of decision, corresponding to the Predictive, Prescriptive, and Optimizing fidelity levels.

Tool life prediction: remaining useful life for each tool in the magazine, updated continuously as sensor data accrues and projected against the upcoming job queue. In the initial deployment, per-product wear contributions are handled through condition-based normalization. Each product gets a wear multiplier derived from a short calibration run, and tool exposure is tracked in equivalent cutting units rather than raw spindle hours. This requires no run-to-failure data and works with mixed-product histories from day one. It does not capture how product sequence or load variation affects wear, but it is deployable before sufficient historical data exists. Once enough labeled wear events have accumulated through normal operation, typically several months of production, the multiplier model can be replaced with a learned encoder without changing the underlying sensor or storage infrastructure.

Preventive maintenance scheduling: identifying optimal intervention windows where tool changes, inspections, and calibrations can be batched without disrupting job flow and order completion, distinguishing between maintenance that can be deferred and maintenance that must be pulled forward to protect a critical run.

Queue optimization: job sequencing that maximizes full order completions within delivery windows while respecting tool life and maintenance constraints.These decision classes are coupled. A tool life prediction changes the feasibility of a queue sequence. A maintenance window reshapes the queue. A queue change alters the rate at which tools are consumed, which shifts the next maintenance window. The twin must model these interdependencies as a system rather than as three independent dashboards. As operational history accumulates, the relationships between products, tools, sequences, and outcomes constitute a knowledge graph that the twin can query to explain not only what is happening but why: sensors track the system’s state, and the knowledge graph tracks the causal structure that produced it.

The Problem/Data/Decision/Stakeholders Matrix (see Figure 1.3) maps each decision pathway to its data requirements and responsible stakeholders. Each row of the matrix reads as a complete decision pathway: the physical problem, the data that bears on it, the decision the data enables, and the stakeholders who act on that decision. The first row works as an example of the shape. Tool Wear Prediction Across Product Types begins with a physical fact about the cell: tools accumulate wear at rates that depend on what they are cutting, so a tool used across multiple products carries a composite wear state that no single product's wear curve describes. The data that bears on this problem is vibration, spindle power, and RPM profiles captured per tool per product, calibrated against empirical wear curves from fresh, mid-life, and end-of-life conditions. The decision those data enable is whether current tool life supports the next queued job, with

replacement timing driven by accumulated wear projected against remaining demand rather than by calendar interval. The stakeholders who act on that decision are maintenance technicians, who execute the replacement, production planners, who schedule around the constraint, and operators, who monitor the in-cycle consequences. The remaining rows follow the same shape at different scales.

The matrix's final row, Sensor Health and Epistemic Integrity, is foundational: every prediction in the rows above it is only as reliable as the sensor layer that row validates.

Problem / Data / Decision / Stakeholders Matrix

Sub-Problem	Problem	Data	Decision	Stakeholders
Tool Wear Prediction Across Product Types	Tools accumulate wear at product-dependent rates. Shared tools carry composite wear states no single product curve describes. Without per-product tracking, replacement defaults to conservative calendar estimates or reactive failure.	Vibration, spindle power draw, RPM profiles per tool per product. Manufacturer life ratings. Empirical wear curves from labeled runs across fresh, mid-life, and end-of-life conditions.	Whether current tool life supports the next queued job. When to replace: by accumulated wear state projected against remaining demand, not by calendar interval.	Maintenance technicians (execute). Production planners (schedule around constraints). Operators (monitor in-cycle).
Preventive Maintenance Scheduling	Calendar or run-hour maintenance replaces tools with remaining life (waste) or misses faster-than-expected degradation (risk). Windows are not coordinated with production demand.	Predicted remaining tool life. Historical failure and replacement records. Upcoming order queue with product types/volumes. Maintenance task durations.	When to open maintenance windows. Which tasks to batch. Whether to pull maintenance forward for high-priority runs or defer through low-criticality segments.	Maintenance technicians (plan/execute). Production planners (coordinate with delivery). Operations management (downtime vs. risk tradeoffs).
Job Queue Optimization	Sequencing affects tool consumption, changeover frequency, and delivery simultaneously. Running similar products reduces changeovers but may delay priority orders. Interleaving balances delivery but potentially accelerates wear. Not simultaneously optimizable without explicit weights.	Active orders (type, quantity, priority, due date). Current tool states. Changeover duration by product pair. Per-product cycle times and consumption rates.	Sequence maximizing full order completions within delivery windows, subject to tool life and maintenance constraints. Requires definition of "optimal": a decision, not a computation.	Production planners (primary). Operations management (priority weights/commitments). Operators (execute).
Real-Time Product Identification	The twin must identify which product is active to select the correct wear model and envelope. Manual logging is unreliable. Without automated classification, downstream models operate on assumed identity.	Vibration, RPM, power draw signatures during machining. Labeled training corpus across products, tool conditions, and blank lots. Classification confidence scores.	Real-time product assignment to model pathway. Confidence-gated: reports uncertainty rather than asserting identity. Below threshold, flags for operator verification.	Digital twin (automated). Operators (verify flags). Maintenance technicians (depend on correct classification for wear attribution).
Performance Envelope Monitoring	Gradual degradation shifts sensor signatures before producing defects or scrap. Current detection is reactive: problems surface as rejected parts or stops, not as early deviations from expected conditions.	Live sensor streams vs. product-specific baselines indexed to tool life stage. Deviation magnitude and rate-of-change metrics.	Whether operation is within bounds for identified product and tool state. Whether deviation warrants a flag (log), alert (notify operator), or halt (prevent scrap/damage).	Operators (respond). Maintenance technicians (diagnose). Production planners (adjust queue if degraded mode).
Sensor Health and Epistemic Integrity	Sensors drift from mounting degradation, cable fatigue, thermal effects. A drifted accelerometer reports a shifted value the model reads as a changed cutting condition. Without cross-validation, degraded sensing is indistinguishable from degraded machining. This is epistemic failure: the system is most confident when it has diverged furthest from reality.	Cross-sensor consistency checks. Baseline drift detection against known-good states. Heartbeat/staleness flags. Historical sensor performance and recalibration records.	Whether to trust model outputs. When to suspend automated decisions and escalate to human verification. When to take sensors offline for recalibration rather than propagate drifted data through downstream models.	Maintenance technicians (recalibrate). Operations management (accept/reject recommendations under uncertainty). All rows above inherit this row's integrity.

Figure 1.3. Problem/Data/Decision/Stakeholders Matrix. Six decision pathways from tool wear prediction through sensor integrity. Read bottom-up for the dependency chain

Scope and Boundaries

The system boundary is the manufacturing cell: blank hopper, two cobots, CNC machine, cleaning station, and finished-part outbox. Upstream supply chain and downstream quality inspection are out of scope for the initial deployment, though both represent interfaces where future expansion would add value.

The temporal scope spans three distinct timescales, and the system must maintain each without collapsing them into a single prediction horizon. Real-time monitoring operates at the cycle level: product identification, envelope compliance, and sensor health assessed continuously during machining. Predictive modeling operates across hours to days: tool life projections and maintenance scheduling that look ahead through the current order queue. Historical analysis operates across weeks to months: wear model refinement, envelope recalibration, and performance trending that improve the twin's accuracy over time.

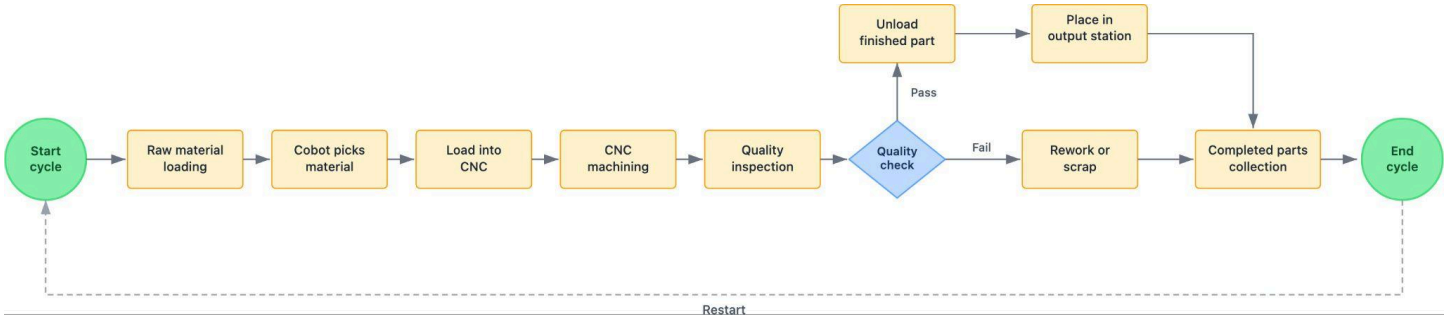


Figure 1.4. Flow chart outlining the manufacturing process end-to-end

Functionally, the twin covers subtractive CNC operations as its primary scope. Robotic arm health is included as a parallel initial-scope track: a failed arm halts the cell as completely as a failed spindle, and UR-series controllers expose joint currents, temperatures, and TCP force via RTDE without additional sensors, supporting anomaly detection using the same architecture applied to the spindle. Arm monitoring at this stage is anomaly detection, not predictive maintenance on arm mechanicals. Coolant system performance and chip evacuation affect cell operation but are excluded from initial scope. If sensor data from these subsystems becomes available through the CNC controller interface and HMI, the architecture should accommodate their inclusion without redesign.

The architecture's modularity reflects a deeper structural choice. The knowledge graph models machines, tools, products, and sensors as typed nodes with typed relationships, and the inference pipeline runs per-tool rather than per-cell. This means the system can be understood as a federation of nested twins: tool-level twins tracking individual wear state and RUL, a machine-level twin managing envelopes, classification, and spindle health, and a cell-level twin coordinating scheduling, cobot status, and queue optimization. The state engine and knowledge graph serve as the coordination layer that lets these nested scopes share data and propagate constraints. Part 4.2 formalizes this structure as a governance framework.

Data: What Is Measurable, Missing, and Uncertain

The sensor suite provides the twin's primary inputs: vibration (accelerometer, per-axis preferred), spindle RPM, spindle and axis motor power draw, and cumulative run time per tool. Additional feeds including coolant flow rate, temperature, and acoustic emission should be evaluated during a sensor audit but are not assumed. The CNC controller exposes data through multiple industrial protocols: OPC UA for bidirectional data read/write, MTConnect for machine monitoring and data collection, EthernetIP for network connectivity,

and PROFINET for servo motor and axis coordination. These provide machine state, active program identification, and potentially tool magazine status. The depth and polling rate of the controller's data export vary by model and vintage. A controller interface audit is a prerequisite before the data architecture can be finalized.

Several critical data needs remain unresolved. Tooling inventory depth is unknown: the number of tools, their overlap across products, and their manufacturer life ratings will determine the complexity of wear modeling and the combinatorial space of the scheduling optimizer. Historical maintenance and replacement records may exist in paper logs, digital systems, or not at all. Their quality and completeness will constrain initial model accuracy.

The most consequential data gap is the absence of labeled training data for the product classifier. Identifying which product is being machined from sensor signatures requires profiles collected across all three products under representative conditions: fresh tooling, mid-life tooling, near-end-of-life tooling, different blank lots. Until this corpus exists, the classifier is speculative. Early-phase operation will require a manual labeling protocol, and the system should report classification confidence rather than assert product identity without qualification.

What Changes

The twin's first contribution is visibility. Using the existing sensors and controller data, the behavioral twin creates a unified, connected view of what the cell is already sensing and doing. This descriptive layer, the phase-zero deliverable, adds value immediately: an operator sees cell state in one place instead of across multiple controller screens, a planner sees job status alongside machine status, a manager sees uptime and cycle counts without requesting manual reports. Everything that follows builds on this foundation.

The value of the digital twin beyond phase zero is the translation between observation and decision. The sensors already exist or can be added. What does not exist is a system that connects those observations to the decisions they should inform.

Today, vibration data exists as a waveform on a controller screen. The twin translates that waveform into a tool life estimate, a maintenance recommendation, and a scheduling constraint. Today, a production planner sequences jobs by due date and intuition. The twin translates tool wear projections and changeover costs into a ranked set of feasible sequences with explicit tradeoffs. A reference dashboard concept (see Figure 1.5) illustrates how these outputs converge in a single operator view, with tool life, queue status, envelope compliance, maintenance windows, and sensor integrity presented together rather than in isolation.

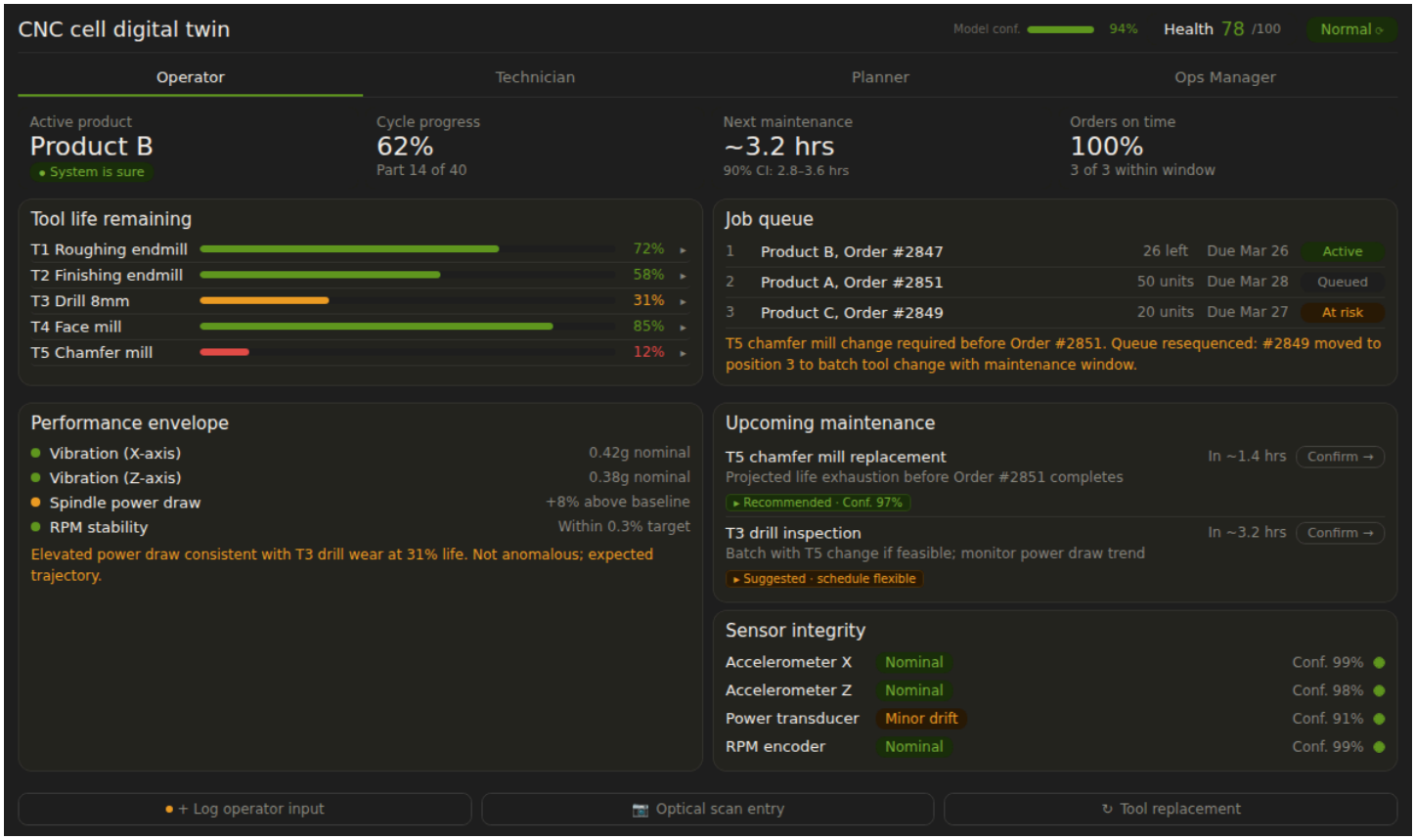


Figure 1.5. Reference dashboard concept. Tool life, job queue, performance envelope, maintenance scheduling, and sensor integrity in a unified operator view.

The efficiency gains are direct: reduced unplanned downtime through predictive rather than calendar-based maintenance, reduced tooling waste through condition-based replacement, and improved on-time delivery through queue optimization that accounts for tool constraints. The safety gain is subtler but arguably more consequential: the system's epistemic integrity layer means that operators and planners know when to trust the twin's recommendations and when to override them.

The learning gain compounds over time. Every operator action feeds back into the model. A confirmed tool replacement is a ground-truth data point. A dismissed alert refines the threshold. A late job completion updates the scheduler. A human override that contradicts the model's prediction is the most valuable training event of all: it reveals exactly where the model's understanding of the physical system was wrong. The twin learns from the consequences of its predictions and from the judgment of the people who act on them. These feedback channels do not improve in isolation: better classification accuracy produces more accurate wear tracking, which produces better-timed maintenance, which produces higher-quality ground truth for the next round of model refinement. The structural uncertainties present at deployment narrow over operational time (Part 3.3 quantifies this convergence).

A system that is confidently wrong is more dangerous than no system at all. The digital twin's value proposition depends on its architecture preventing that failure mode. Every design decision in the sections that follow will be evaluated against that standard.

Section 2: Physical System Characterization & Data Acquisition

Part 2.1: Physical System Characterization

The cell is built around a TRAK VMC7si vertical machining center controlled by a Siemens SINUMERIK ONE CNC. The machine provides 30" × 20" × 20" of travel (X, Y, Z), a 12,000 RPM spindle with hybrid ceramic bearings, a 24-station automatic tool changer with dual-arm exchange, and high-pressure coolant through spindle. Two Universal Robots cobots manage blank loading from a hopper, finished part extraction, preliminary cleaning, and outbox placement. The cell footprint, including the chip conveyor and coolant tank, is approximately 144" × 112".

Figure 2.1 Inspiration of the physical setup of the automated manufacturing cell - front view

Figure 2.2 Inspiration of the physical setup of the automated manufacturing cell - side view

Physical Processes and Measurable Phenomena

When the spindle is running and the tool is in the cut, the cell announces itself. The sound shifts as the flute engages aluminum: a rising pitch under load, a change in the coolant spray pattern, a faint thermal shimmer above the workpiece. These are the phenomena the machinist reads by instinct and the digital twin must learn to read by instrument.

Material removal is the cell's primary physical process. Part geometry is defined in CAD, converted to G-code through CAM, and verified through a dry run before the cell enters automated mode with cobot material handling. In operation, a rotating tool engages a stationary aluminum workpiece, and the cutting edge shears material at the tool-chip interface. This engagement generates three coupled phenomena the digital twin must observe: cutting forces, vibration, and heat.

Cutting forces arise from the workpiece material's mechanical resistance to the tool's motion, loading the spindle and axis drive motors. The SINAMICS S120 drive system reports the resulting changes in motor current and torque continuously. As a tool wears, its force signature shifts in characteristic ways: flank wear

increases the contact area and raises friction forces, crater wear alters chip flow, and built-up edge formation changes the effective geometry unpredictably. These shifts are what the wear model must learn to detect.

Vibration is the dynamic expression of cutting forces. Each flute's engagement produces a periodic vibration whose frequency depends on spindle speed and flute count, typically 300 to 800 Hz for this machine's aluminum operations. Harmonics extend the useful diagnostic range to roughly 3 kHz. As a tool wears, its vibration signature broadens and intensifies. Each product's unique tool path creates a characteristic vibration profile: the physical basis for the product classifier.

Heat generation at the cutting zone is proportional to the power consumed in material removal. The SINUMERIK ONE's thermal compensation system corrects for structural drift, but tool-side thermal accumulation contributes to wear progression. Coolant flow rate and pressure modulate this thermal balance.

The cobots' physical processes are simpler but operationally critical. Each load/unload cycle subjects the arm joints to predictable torque profiles. The UR controllers report joint currents, temperatures, and TCP force via RTDE at up to 500 Hz. Deviations from expected torque profiles indicate gripper degradation, fixture misalignment, or mechanical wear.

Key Performance Variables and Units

The system's measurable state spans four domains: cutting dynamics (spindle torque, axis torques, spindle power, vibration acceleration), kinematics (spindle speed, axis positions, velocities, feed rate override), thermal (motor temperatures, coolant flow rate, coolant temperature), and cell state (active NC program, tool in spindle, tool measurements, cycle status, cobot joint currents, cobot TCP force). The complete variable inventory with units, sampling frequencies, acquisition tiers, and data ownership appears in the tables in Part 2.3.

Temporal and Spatial Scales

The physical phenomena span five orders of magnitude in time. Tooth engagement occurs at intervals of 1–3 ms, setting the minimum sampling rate for vibration capture. A machining cycle for one part runs minutes. A tool's useful life spans hours to tens of hours depending on the product mix. Wear model refinement and envelope recalibration operate across weeks to months as ground-truth data accumulates. The digital twin must maintain data streams at each of these timescales without collapsing them: kHz-rate raw vibration for the classifier, Hz-rate drive telemetry for envelope monitoring, per-cycle summaries for the wear model, and per-replacement records for long-term model refinement.

Spatially, the critical phenomena concentrate at the tool-workpiece interface, a contact zone measured in millimeters, but their observable signatures propagate to the spindle housing, the drive motors, and the controller. The sensing strategy exploits this propagation: the twin does not observe the cutting zone directly but infers its state from signals that have traveled through the machine's mechanical and electrical structure.

Part 2.2: Sensing and Data Acquisition

Three-Tier Sensor Architecture

The sensing strategy is organized into three tiers, distinguished by data routing and hardware requirements. This tiering reflects the boundary between what the machine already knows about itself and what must be added for the twin to function.

Tier 1: Controller-native data. The SINUMERIK ONE with integrated SINAMICS S120 drive system and S7-1500F PLC continuously monitors spindle and axis motor currents, torques, power consumption, and temperatures. These values exist because the drive system needs them for its own closed-loop control; the twin accesses them as a secondary consumer via OPC UA. The controller also exposes axis positions, velocities, following error, spindle RPM, active NC program, tool in spindle, magazine state, cycle status, feed rate override, and alarm history. This is the richest data tier and requires no additional hardware.

Tier 2: Retrofit, low-frequency. Sensors with standard industrial outputs (4–20 mA, 0–10 V, IO-Link) connect directly to the S7-1500F PLC's analog or digital input modules. Candidate additions include a coolant flow rate sensor, coolant concentration sensor and ambient temperature probes, and a conditioned accelerometer outputting RMS vibration level. These route through the existing PLC and are accessible via the same OPC UA server as Tier 1 data.

Tier 3: Enhanced vibration monitoring (if required). If Tier 1 drive signals and Tier 2 conditioned vibration prove insufficient for product classification or spectral wear analysis, the architecture accommodates an industrial condition monitoring module such as the Siemens CMS1281. Unlike a research-grade DAQ, the CMS1281 is designed for the S7 ecosystem: it accepts IEPE accelerometer inputs, samples at kHz rates internally, and delivers frequency-domain features (band energy, spectral peaks, envelope spectra) as

structured data accessible through the Siemens infrastructure. The distinction between Tier 2 and Tier 3 vibration is diagnostic depth: Tier 2 provides a summary metric (RMS level, indicating how much the machine is vibrating), while Tier 3 provides spectral decomposition (frequency-band content, indicating how it is vibrating and enabling discrimination between wear modes, product signatures, and chatter onset). Whether this depth is needed is an empirical question the calibration phase will answer. The architecture supports adding Tier 3 without redesign.

In-situ tool inspection. An on-machine tool setter (Renishaw TRS non-contact laser or contact-based probe, mounted on the CNC table) provides automated, in-process measurements of tool length and effective diameter without requiring the removal of the tool from the spindle. Probing cycles can be programmed into the workflow at defined intervals: every n^{th} part, every tool change, or at start of shift. Each measurement is written to the SINUMERIK ONE's tool management system and is accessible via OPC UA as Tier 1 data. The digital twin consumes each probing result as a label data point with tool ID, timestamp, measured length, measured diameter, and cumulative job history since the previous measurement. Progressive shortening and diameter reduction correlate with wear accumulation. Sudden changes indicate chipping or breakage. This process provides a reasonably continuous automated calibration signal for the wear model from day one, with zero additional operator workload.

In-situ probing measures macro geometry (length, effective diameter) rather than micro geometry (flank wear, VBmax, crater wear). The macro measurements track the trajectory of tool degradation and detect breakage events, but they do not directly quantify the wear features that determine remaining useful life. The relationship between macro and micro geometry must be established through offline measurement as described in the following section.

Calibration-phase ground truth: periodic tool scanning. During the initial calibration period (~3-6months), periodic offline tool measurement provides direct ground-truth labels for the wear models that no streaming sensor or in-situ probe can provide. 3D profilometry or optical inspection of a removed tool occurs at a resolution fine enough to characterize the actual condition of the cutting edge. Each such measurement enters the system as an operator-entered labeled record including tool ID, timestamp, VBmax value, wear photograph or scan file, and the tool's full job history since installation or last measurement. This may result in an increased workload for operators and technicians during the calibration phase, but will taper to only periodic validation measurements after initial calibration is complete. Scan results enter the system as structured operator input events. As such, they are stored in the SQLite database alongside maintenance and replacement records.

Offline measurements serve two roles: 1) establishing the functional mapping between in-situ tool readings and actual wear state, capturing and calibrating the shape of the wear curve between installation and replacement.; 2) continued periodic measurements validate that the mapping remains accurate even as tooling, materials, or cutting programs change.

If the combination of Tier 1 drive signals, in-situ probing, and offline calibration data proves sufficient for wear model accuracy, Tier 3 enhanced vibration monitoring may never need to be deployed.

The behavioral twin runs entirely on Tier 1 data. Whether the predictive twin requires Tier 2 and Tier 3, or whether drive-level torque and current signatures prove sufficient for product classification and wear detection, is an empirical question the calibration phase will answer. The architecture supports both outcomes without redesign. The cell layout with sensor placements for all three tiers appears in Figure 2.3.

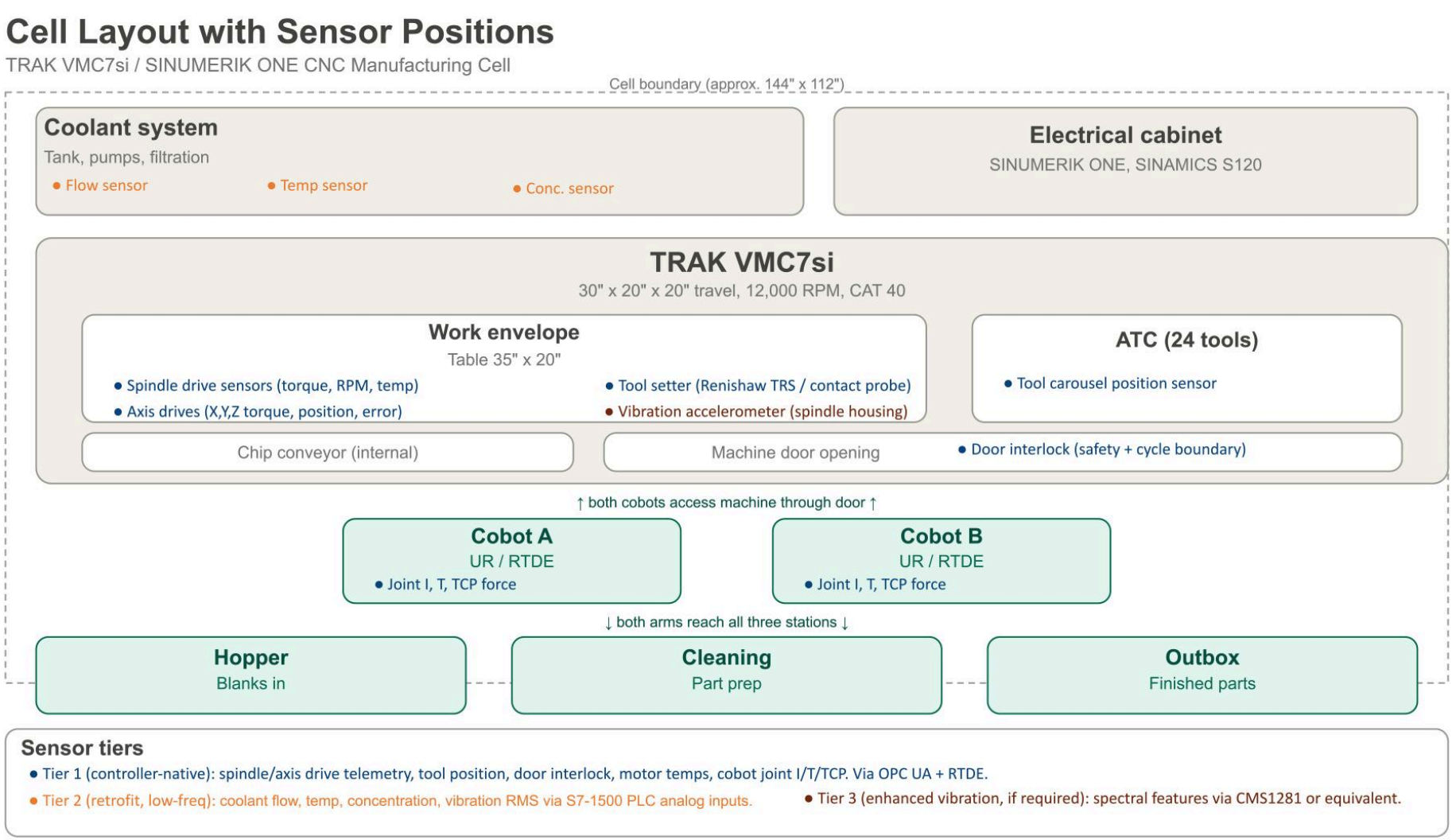


Figure 2.3. Top-down cell layout showing TRAK VMC7si, cobot positions, hopper, outbox, and sensor locations. Tier 1 sensors (controller-native) shown in blue; Tier 2 retrofit positions shown in amber, Tier 3 in red.

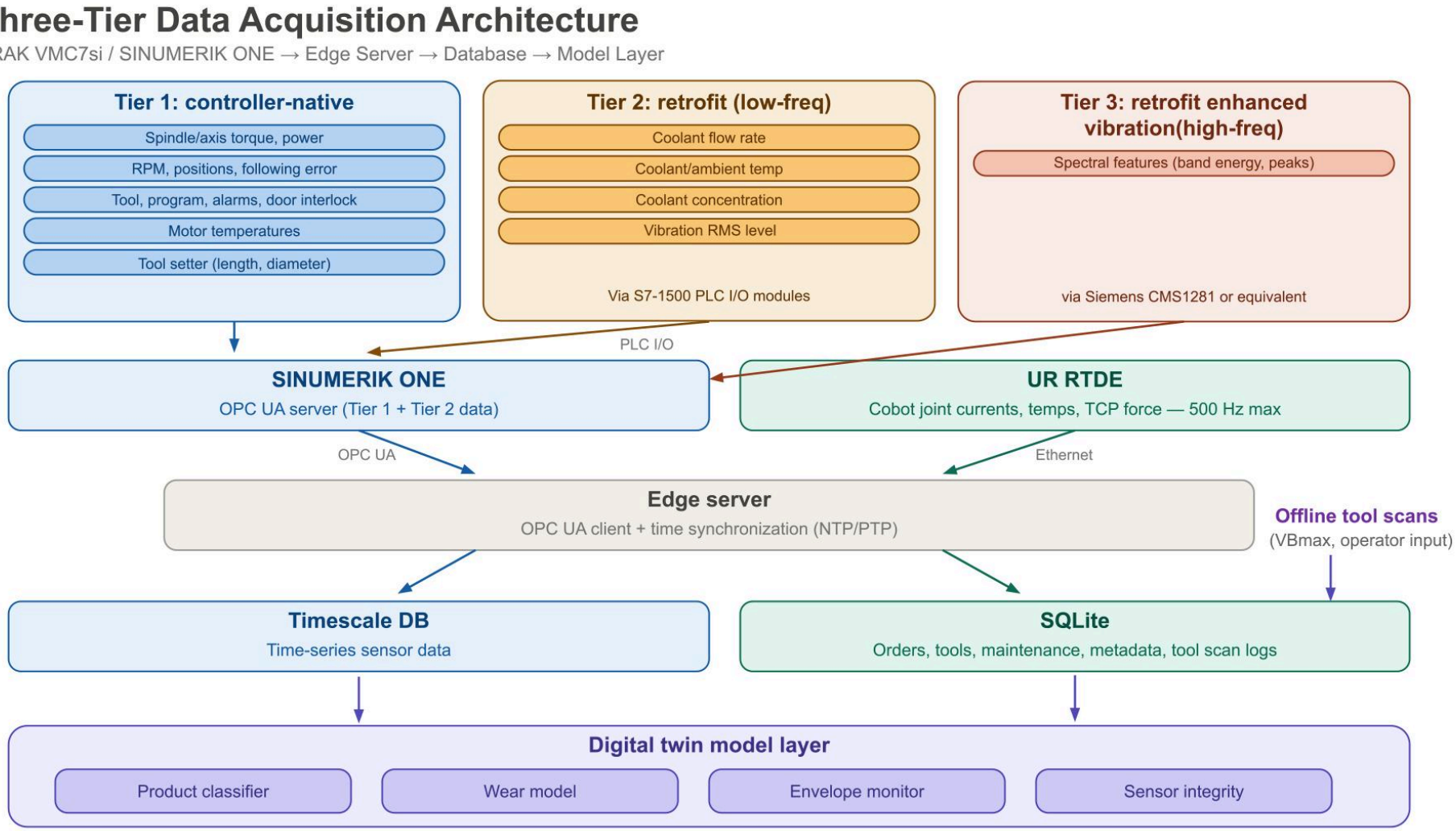


Figure 2.4. Data acquisition architecture. Three tiers converge in the time-series database. Tier 1 (controller-native) and Tier 2 (retrofit low-frequency) route through OPC UA. Tier 3 (enhanced vibration, if deployed) integrates through the Siemens ecosystem. All streams are time-synchronized.

Sampling Rates and Justification

Tier 1 drive data is available at the servo cycle rate, typically 250 Hz to 1 kHz. Axis positions update at 125–250 Hz. Tool and program state variables are event-driven. Cobot RTDE data streams at up to 500 Hz. Tier 2 process sensors sample at 1–10 Hz, consistent with the timescales of thermal phenomena. The sampling rates for all variables appear in the tables in Part 2.3.

If Tier 3 enhanced vibration monitoring is deployed, spectral analysis requires sampling rates in the 10–20 kHz range. This is set by the Nyquist requirement for the tooth-passing frequency and its harmonics: at 12,000 RPM with a 4-flute endmill, the fundamental TPF is 800 Hz, and its fourth harmonic extends to 3,200 Hz, requiring a minimum sampling rate of 6,400 Hz. The 10–20 kHz range provides margin for higher-order harmonics, structural resonance content, and anti-aliasing filter roll-off. An industrial condition monitoring module such as the Siemens CMS1281 handles this sampling internally and delivers process spectral features as a reduced output rate.

Data Path and Integration

All three tiers converge in the database layer (see Figure 2.4). Tier 1 and Tier 2 data exit through the SINUMERIK ONE's OPC UA server; an OPC UA client on the edge server subscribes to the relevant nodes and writes to an TimescaleDB time-series database. In-situ tool setter measurements flow through the same Tier 1 path: the probing cycle writes results to the controller’s tool management system, and the OPC UA client captures them alongside drive telemetry. Offline tool scan results (VBmax measurements from 3D profilometry or optical inspection) enter through the dashboard as structured operator input events and are written to the SQLite relational database alongside maintenance and replacement records. Machine metadata (tool catalog, order queue, maintenance records, tool scan logs) also resides in SQLite. If Tier 3 enhanced vibration is deployed, the CMS1281 integrates with the Siemens ecosystem and its spectral features are accessible through the same OPC UA infrastructure as Tier 1 and Tier 2 data. Synchronized timestamps across all tiers enable the model layer to query any combination of controller state, process variables, tool geometry measurements, wear ground truth, and vibration features for a given time window. Figure 2.5 shows which variables feed which of the four model components described in Part 3.1.

Data Quality and Uncertainty

Tier 1 data inherits the accuracy of the drive system's own calibrated sensors. The primary quality risk is not measurement error but interpretation ambiguity: a change in spindle torque could reflect changing cutting conditions, changing tool state, or changing workpiece material properties. Tier 2 and Tier 3 retrofit sensors introduce calibration drift, mounting resonance effects, and fouling. Consider the consequence chain for a single Tier 2 failure: a coolant flow sensor fouls gradually over weeks. The reported flow reading drifts downward. The envelope monitor, seeing reduced flow, may widen its acceptable vibration range to account for expected thermal effects. The wear model may adjust its thermal contribution estimate. But the physical coolant flow has not changed; the sensor has drifted. The consequence has crossed from the physical domain to the informational domain: operators and automated systems are now making adjustments to compensate for a condition that does not exist. This is the physical-to-informational crossover pattern that the epistemic integrity layer, described in Part 3.1, is designed to detect. How the model layer handles these quality challenges, including cross-sensor consistency checks, staleness tracking, feature extraction, and normalization strategy, is addressed there as part of the data assimilation architecture.

One normalization decision belongs here because it shapes the data acquisition design: the product classifier and the wear model receive the same vibration features but normalized differently. The classifier scales each

feature relative to its distribution under known-good cutting conditions for each product type, ensuring response to signature shape rather than absolute amplitude. The wear model receives unnormalized features, since absolute amplitude changes are what it needs to track. This dual normalization requires storing both the raw feature values and the per-product-type reference distributions in the database layer.

Part 2.3: Variable and Data Tables

The following tables summarize the measurable variables for the CNC subsystem and the cobot subsystem. Each variable is assigned to its acquisition tier, linking back to the data path described in Part 2.2.

CNC Subsystem Variables

Variable	Description	Sensor / Source	Units	Freq.	Tier	Data Owner
Spindle torque	Instantaneous spindle motor torque	SINAMICS S120 drive (motor current × torque constant)	Nm	250 Hz–1 kHz	Tier 1	CNC Controller
Spindle power	Active power consumed by spindle motor	SINAMICS S120 drive (computed from V × I)	kW	250 Hz–1 kHz	Tier 1	CNC Controller
Spindle RPM	Spindle rotational speed	SINAMICS S120 encoder feedback	RPM	250 Hz–1 kHz	Tier 1	CNC Controller
Axis torques (X,Y,Z)	Feed axis motor torques	SINAMICS S120 axis drives	Nm	250 Hz–1 kHz	Tier 1	CNC Controller
Axis positions (X,Y,Z)	Current axis positions	SINUMERIK ONE position feedback (rotary encoders on ballscrews)	mm	125–250 Hz	Tier 1	CNC Controller
Feed rate override	Operator or RTAC-adjusted feed rate as % of programmed	SINUMERIK ONE NC variable	%	Event	Tier 1	CNC Controller
Motor temps	Spindle and axis motor winding temps	Drive-integrated thermal sensors	°C	1 Hz	Tier 1	CNC Controller
Active NC program.	Currently executing program identifier	SINUMERIK ONE NC variable	ID	Event	Tier 1	CNC Controller
Tool in spindle	Magazine position of active tool	ATC / tool management	Pos #	Event	Tier 1	CNC Controller
Cycle status	Machine state: idle, cutting, rapid traverse, tool change, alarm	SINUMERIK ONE PLC	Enum	Event	Tier 1	CNC Controller
Alarm history	Active and historical alarm codes with timestamps	SINUMERIK ONE alarm system	Code	Event	Tier 1	CNC Controller
Door interlock	Machine door open/closed state; serves as safety interlock and cycle boundary marker	Safety PLC discrete input (S7-1500F)	Bool	Event	Tier 1	CNC Controller
Tool length	Measured tool length from in-situ probing cycle	Renishaw TRS or contact tool setter via SINUMERIK ONE probing interface	mm	Per probing cycle	Tier 1	CNC Controller
Tool diameter	Measured effective tool diameter from in-situ probing cycle	Renishaw TRS or contact tool setter via SINUMERIK ONE probing interface	mm	Per probing cycle	Tier 1	CNC Controller
Coolant concentration	Coolant mixture ratio	Retrofit concentration sensor	%	1 Hz	Tier 2	Maint. Team
Coolant flow rate	Volume flow through spindle and flood coolant	Retrofit flow sensor (4–20 mA)	L/min	1 Hz	Tier 2	Maint. Team
Coolant temp.	Coolant supply temperature	Retrofit RTD or thermocouple	°C	1 Hz	Tier 2	Maint. Team
Vibration (RMS)	Broadband vibration level at spindle housing	Retrofit accelerometer (conditioned output)	g	10 Hz	Tier 2	Maint. Team
Vibration (spectral)	Frequency-domain features for spectral analysis	Industrial condition monitoring module (e.g. Siemens CMS1281)	band energy (g^2), spectral peaks (Hz, g), envelope RMS (g)	Processed internally at 10–20 kHz	Tier 3	DT Engineering
Tool wear (VBmax)	Flank wear measurement from offline 3D profilometry or optical inspection	Operator input via dashboard (offline measurement)	mm	Per scan event	Operator input	Maint. Team

Cobot Subsystem Variables

Variable	Description	Sensor / Source	Units	Freq.	Tier	Data Owner
Joint currents (J0–J5)	Motor current per joint during manipulation cycle	UR RTDE interface	A	≤500 Hz	Tier 1	CNC Controller
Joint temperatures	Motor temperature per joint	UR RTDE interface	°C	≤500 Hz	Tier 1	CNC Controller
TCP force/torque	Force and torque at tool center point (6-DOF)	UR RTDE (built-in F/T estimation)	N, Nm	≤500 Hz	Tier 1	CNC Controller
Robot mode	Operating state: running, idle, protective stop, fault	UR RTDE interface	Enum	Event	Tier 1	CNC Controller
Safety status	Safety system state: normal, reduced, safeguard stop	UR RTDE interface	Enum	Event	Tier 1	CNC Controller
Program state	Active cobot program and execution step	UR RTDE interface	ID	Event	Tier 1	CNC Controller
Actual TCP pose	Tool center point position and orientation (6-DOF)	UR RTDE interface	mm, rad	≤500 Hz	Tier 1	CNC Controller

All Tier 1 CNC variables are accessible via the SINUMERIK ONE OPC UA server, including in-situ tool setter measurements which flow through the controller's probing interface. The door interlock serves dual duty as a safety input and a cycle boundary marker for segmenting sensor data into per-part windows. All cobot variables are accessible via the UR RTDE protocol. Tier 2 variables route through the S7-1500F PLC's I/O modules and are then accessible via the same OPC UA server. If Tier 3 enhanced vibration is deployed, the CMS1281 or equivalent module integrates with the Siemens ecosystem and its spectral features are accessible through OPC UA. Offline tool wear measurements (VBmax from 3D profilometry or optical inspection) enter as operator input events through the dashboard and are stored in the relational database. The data ownership column identifies which team is responsible for sensor maintenance, calibration, and data quality assurance for each variable.

Variable-to-Model Data Flow

Which variables feed which model components in the digital twin engine

Source	Variable / group	Product classifier	Wear model	Envelope monitor	Sensor integrity
Tier 1	Spindle torque	●	●	●	●
	Spindle power		●	●	●
	Spindle RPM	●		●	●
	Axis torques (X, Y, Z)	●		●	●
	Axis positions / velocities	●		●	
	Feed rate override			●	
	Motor temperatures			●	●
	Active NC program	●			
	Tool in spindle (ATC position)	●	●		
	Cycle status		●	●	●
	Alarm history			●	
	Door interlock (cycle boundary)		●		
	Tool length (in-situ setter)		●		
	Tool diameter (in-situ setter)		●		
Tier 2	Coolant flow rate			●	●
	Coolant temperature			●	●
	Coolant concentration			●	●
	Vibration RMS	●	●	●	●
Tier 3	Vibration spectral features	●	●		●
Cobot	Joint currents (J0–J5)			●	●
	Joint temperatures			●	●
	TCP force/torque			●	●
	Robot mode / safety status			●	
Operator input	Tool Wear (VBmax, offline scan)		●		

Note: Door interlock segments continuous sensor streams into per-part windows. All four models depend on this segmentation for per-cycle feature extraction.

Figure 2.5. Data flow from variable sources through acquisition tiers to the four model components (product classifier, wear model, envelope monitor, sensor integrity). Circle markers indicate which variables feed which models.

Section 3: Virtual Representation and Analytics Layer

Part 3.1: Model Plan

The data acquisition architecture described in Part 2.2 delivers synchronized sensor streams from three tiers into a unified database layer. This section describes the model that consumes those streams: its components, the logic that connects them, and the mechanisms by which it learns from operational outcomes.

3.1.1 Model Architecture

Hierarchy and Primary Components

The digital twin's model layer is organized as a four-layer stack: physical data acquisition, ML inference, operational scheduling, and stakeholder presentation. Each layer communicates with adjacent layers through typed interfaces. No layer reaches across the stack. A central state engine coordinates ML inference, alert evaluation, and downstream scheduling, serving as the sole cross-layer authority.

The ML inference layer implements the four components defined in Part 1 (product classifier, wear model, envelope monitor, sensor integrity). This section describes how each is built and where uncertainty accumulates.

The *product classifier* is implemented as a sequence classifier trained on the TUAWS aluminum cutting dataset (99.93% validation accuracy). Its output selects the wear multiplier applied to every subsequent RUL estimate. As noted in Part 1, a misclassification does not add noise; it selects a different wear trajectory entirely.

The *wear model* has two implementations: TwinnableTFT, a Temporal Fusion Transformer with a Variable Selection Network and frozen backbone shared across six industrial domains, handles the primary RUL regression. A standalone Stack-LSTM trained on the IEEE NUAA Ideahouse dataset handles CNC-specific tool wear prediction from 300 Hz milling signals. Both export to ONNX for CPU inference at under 100 ms per tick.

These components form a dependency chain: classifier output feeds multiplier selection, which modulates the RUL estimate, which feeds the scheduler's deadline constraint. The sensor integrity module cross-cuts all of them, gating every prediction with a data-quality score.

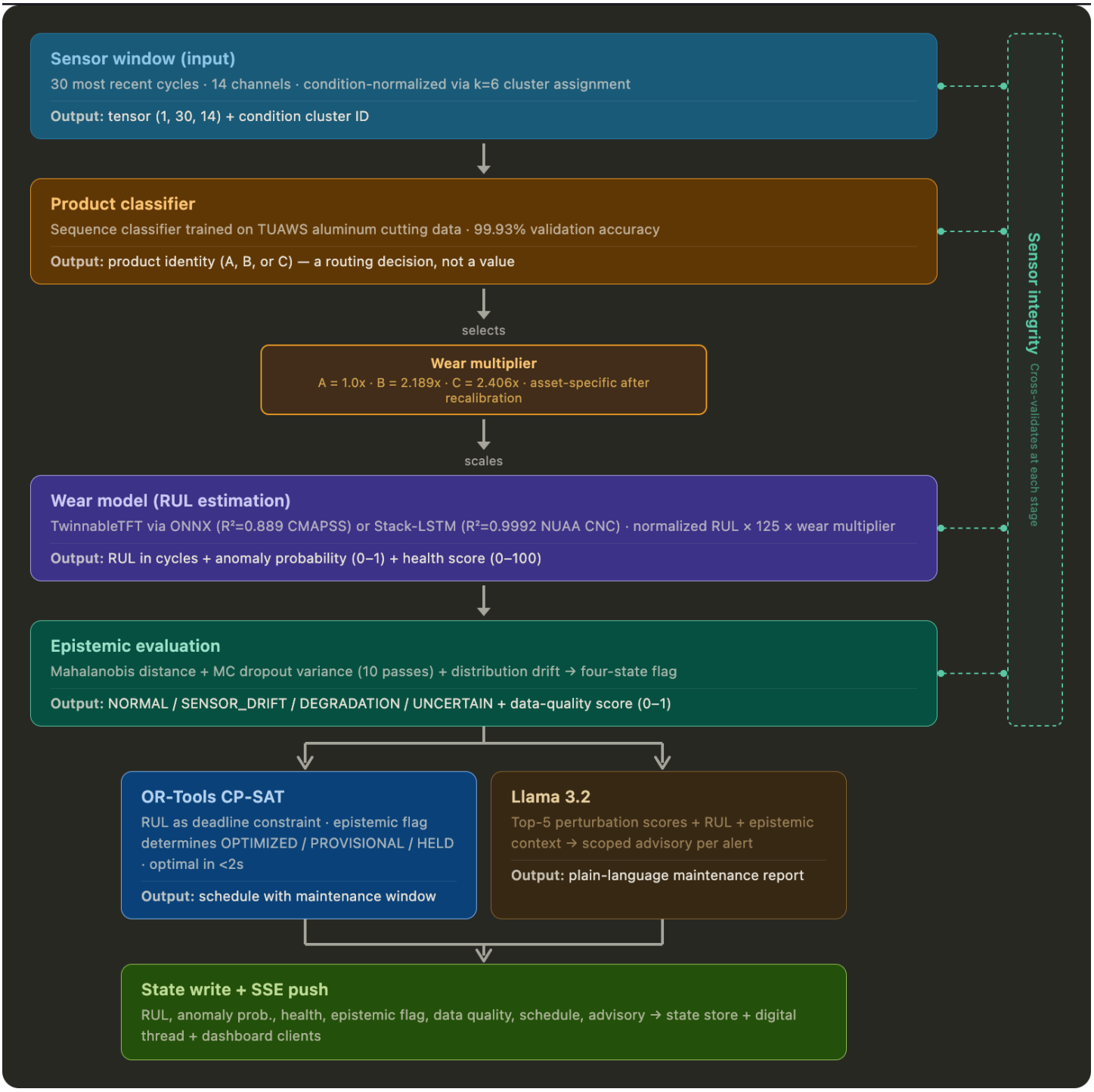


Figure 3.1. End-to-end inference pipeline. A 30-cycle sensor window feeds a product classifier whose A/B/C output selects a wear multiplier, which scales the RUL estimate from the wear model. An epistemic evaluation stage assigns a four-state integrity flag and data-quality score. The pipeline then forks: OR-Tools CP-SAT produces a maintenance schedule constrained by RUL and the epistemic flag, while Llama 3.2 generates a plain-language advisory. Both branches converge at the state write for persistence and dashboard push. A sensor integrity module (dashed) cross-validates at each stage.

Physical-to-Digital Connection

The physical-to-digital connection operates as a push pipeline at the production cycle boundary. The CNC controller exposes cycle-aggregated signals through the OPC UA server described in Part 2.2. An edge agent reads these signals at end-of-cycle and publishes a structured record to the ingestion pipeline. The FastAPI

backend writes each record to TimescaleDB's sensor_readings hypertable immediately. In demonstration mode, a shaped 192-cycle synthetic replay sequence substitutes for live sensor data; the inference pipeline is identical, and only the data source changes.

Synchronization is maintained by the state engine tick loop. Every tick (default interval: 30 seconds, configurable), the state engine queries TimescaleDB for the most recent 30 cycles, constructs the input tensor, runs inference through the active domain head, evaluates alert conditions, updates the local state store, pushes the updated state to all connected dashboard clients via server-sent events, and writes a causal event to the digital thread. The pipeline is single-threaded within the tick, eliminating race conditions between inference and alert evaluation.

The return path closes the loop. The five operator input types defined in Part 1 (maintenance confirmation, anomaly acknowledgment, job completion report, machine status override, and shift context) each have a dedicated API endpoint, data schema, and downstream effect on the model's state.

Feedback Mechanisms

Three nested feedback loops operate at distinct timescales.

Cycle-level (seconds). Sensor readings enter the state engine on every tick. The RUL estimate directly drives the OR-Tools maintenance deadline constraint: a tighter RUL produces a tighter constraint, which changes the optimal job sequence immediately. Alert thresholds are hot-reloadable; a change takes effect on the next tick with no system restart. This is the fastest feedback path: physical state change to model update to schedule update in under one tick interval.

Shift-level (hours). Five operator feedback channels are captured via dedicated API endpoints and written to the local state store and digital thread. Maintenance confirmation immediately resets the live RUL counter. Machine status override immediately triggers a schedule replan. Job completion data and anomaly acknowledgments accumulate for future model improvement. A shift-close event snapshots the full system state for trend analysis.

Recalibration (days to weeks). As the deployed system accumulates tool replacement events, the per-product wear multipliers can be recomputed from observed data. When at least three replacement events per product type exist, the recalibration endpoint replaces the prior values (derived from the UC Berkeley milling dataset: Product A = 1.0×, Product B = 2.189×, Product C = 2.406×) with multipliers computed from the actual equivalent cutting units at replacement observed on this specific machine. After sufficient deployment, the RUL estimates reflect the specific material, spindle, and tooling combination of the installed cell, not a generic benchmark.

The feedback architecture is designed to narrow the structural uncertainties identified in the system's initial deployment state. The classifier's real-world accuracy becomes empirically measurable within weeks as operators verify product identity at changeover. The RUL model's prediction error becomes measurable per replacement event. The wear multipliers converge toward asset-specific values over months. The rate of convergence is set by the slowest-accumulating channel: multiplier recalibration, which needs product-segmented replacement data across months of operation. How these convergence rates are monitored and what happens when they stall is addressed in Part 4.1. One structural risk in this feedback architecture requires explicit treatment: censored data. When operators replace tools before the model's predicted failure point, the system never observes the outcome it was predicting. If conservative replacements dominate the training data, the system learns a biased picture skewed toward early intervention, which reinforces the operator behavior that generated the bias. The loop is self-perpetuating and looks like stability while concealing the information the system most needs. 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. The feedback rate is also self-regulating in a way that benefits convergence. When predictions are confident and correct, operators follow recommendations, generating expected outcomes. When the system is uncertain, operators apply their own judgment, generating overrides concentrated at the frontier where the model's uncertainty is genuine and additional evidence has the highest value. Early learning is fast, when everything is uncertain and overrides are frequent, tapering to targeted learning at the model's genuine uncertainty frontier.

3.1.2 Data Assimilation and Updating

Real-Time Data Refresh

The state engine reads a 30-cycle sliding window from the three-tier sensor architecture (Part 2.2) on every tick. The window slides forward by one cycle, maintaining a fixed-length recent history. The window size of 30 was determined empirically as the minimum that captures a complete degradation signature.

State variables are updated at the end of each tick. The state engine writes the current RUL estimate, anomaly probability, health score, epistemic flag, data quality score, active domain, and timestamp to the local state store. Only the most recent row is actively consumed; historical rows form the twin state timeline that the dashboard renders as a rolling RUL chart.

Calibration

The sliding-window inference pipeline estimates tool condition from indirect sensor signatures. Part 2.2 describes the two direct measurement paths that provide ground truth: in-situ tool probing and offline calibration scans.

The model consumes these ground-truth paths as label data. Each in-situ measurement enters as a calibration point keyed to the tool's cumulative job history, allowing the wear model to check its indirect estimates against direct measurement. Where the two diverge, the divergence is itself a diagnostic signal: either the feature-to-wear mapping needs updating, or the cutting conditions have shifted outside the training distribution. Offline VBmax measurements establish the functional relationship between the macro geometry the tool setter measures and the micro geometry that determines remaining useful life.

Handling Missing, Delayed, and Noisy Data

The sliding window query requires exactly 30 rows. If fewer than 30 exist (cold start, reset, or gap in sensor data), the state engine enters a warming-up state and suppresses all predictions. The dashboard displays a data quality indicator and no alerts fire. This prevents the model from operating on an incomplete window.

Delayed data is handled by TimescaleDB's chunk-based architecture, which tolerates out-of-order inserts. The state engine queries by descending timestamp, so late-arriving records enter a future window once they arrive. A configurable maximum latency parameter filters out records older than a threshold, preventing stale data from re-entering the window.

Noisy data is addressed at three levels. The condition-aware StandardScaler normalizes sensor values relative to the expected range for the current operating condition, preventing the model from conflating condition-driven variation with degradation-driven variation. The Variable Selection Network learns to gate each input feature's contribution per timestep, effectively down-weighting uninformative sensors. Sensors identified as constant across the training set (standard deviation below 0.01) were removed at the feature engineering stage.

Cross-Sensor Consistency and Staleness Tracking

The model addresses interpretation ambiguity (Part 2.2) through cross-validation between independent signals. When one feed diverges while the others agree during a known cutting condition, the sensor integrity module flags the discrepancy before the error propagates.

Each variable carries a freshness timestamp. If an update is not received within a configurable window (typically 5× the expected sample interval), the variable is flagged stale and downstream models widen their confidence intervals or revert to last-known-good estimates. The system never silently uses stale data as current. The boundary of this approach, identified in Part 1, is correlated drift: when all sensors shift together, cross-validation has no disagreeing signal to detect. That case is handled through confidence interval widening and escalation to human judgment.

Condition Normalization

The model uses condition-aware normalization as its primary calibration mechanism. During training, k-means clustering over operational settings features groups operating cycles into k = 6 condition clusters. A separate StandardScaler is fitted per cluster. At inference time, the incoming window is assigned to the nearest centroid and scaled by the corresponding scaler. This prevents the model from conflating condition-driven sensor variation with degradation-driven variation. The dual normalization strategy for the classifier and wear model (shape-normalized vs. unnormalized features) is described in Part 2.2.

Epistemic Integrity as Continuous Calibration

The epistemic integrity layer (Part 1) runs as a continuous calibration check on every tick. Three signals feed it: Mahalanobis distance from the training distribution, MC dropout variance (10 forward passes), and distribution drift detection. Together these produce the four-state flag using threshold rules on the combined signals.

The epistemic layer cannot detect its own domain gap: predictions may be precise by the model's own assessment yet systematically offset from reality. How long the feedback channels take to close that gap depends on the rate at which replacement events accumulate, and on whether the operating conditions during that accumulation period are representative of the conditions the model will face afterward.

3.1.3 Model Logic and Expected Outputs

Inference Pipeline

One complete inference cycle proceeds through the following logical sequence. Each step produces a defined output that feeds the next.

1. *Window assembly.* Query the sensor readings table for the most recent 30 rows. Each row contains 14 sensor channels (7 constant sensors removed during feature engineering). Construct the input tensor of shape (1, 30, 14).

2. *Condition detection.* Extract operational settings features from the window. Compute distances to the k = 6 condition centroids fitted during training. Assign the window to the nearest centroid. Apply the per-condition StandardScaler to normalize the input.

3. *ONNX inference.* Pass the normalized input through TwinnableTFT via ONNX Runtime on CPU. The model applies the Variable Selection Network (producing attention weights per sensor per timestep), the 2-layer Transformer encoder (shared frozen backbone), and the active domain head. Outputs: normalized RUL $\in [0, 1]$ and anomaly probability $\in [0, 1]$.

4. *Denormalization.* For the CMAPSS domain: RUL in cycles = normalized RUL × 125, where 125 is the piecewise linear target cap. For the CNC domain: the Stack-LSTM produces RUL in cutting seconds, denormalized by run duration.

5. *Product classification and wear multiplier.* The product classifier assigns the current sensor window to one of three product types. The corresponding wear multiplier (Product A = 1.0×, B = 2.189×, C = 2.406× at initial deployment, replaced by asset-specific values after recalibration) scales the base RUL estimate.

6. *Epistemic evaluation.* Compute Mahalanobis distance from the training distribution. Run 10 MC dropout forward passes and compute variance. Compute drift score from rolling distribution statistics. Classify into the four-state flag.

7. *Health score.* health = clip((normalized RUL × 100) − (anomaly probability × 20), 0, 100). This maps RUL and anomaly probability to a single 0–100 indicator for the Operator view.

8. *Alert evaluation.* If RUL in cycles ≤ alert threshold (default: 30) and sufficient cycles have elapsed since the last alert (cooldown: 20 cycles): fire alert. Write to the maintenance log. Trigger OR-Tools scheduling. Call Llama 3.2 with the top-5 perturbation sensitivity scores and domain context to generate a plain-language maintenance advisory.

9. *State write.* Write RUL, anomaly probability, health, epistemic flag, data quality score, domain, and timestamp to the local state store. Push a server-sent event to all connected dashboard clients. Write a causal event to the digital thread.

Key Formulas

Quantity	Formula / Method	Output Range
RUL (cycles)	TwinnableTFT ONNX → normalized RUL × 125	0–125 cycles
Anomaly probability	Sigmoid output of domain head binary classifier	0.0–1.0
Health score	clip(normalized RUL × 100 − anomaly probability × 20, 0, 100)	0–100
Mahalanobis distance	$d = \sqrt{((x-\mu)^T \Sigma^{-1} (x-\mu))}$ against training distribution	≥ 0; > 3.5 → anomalous
MC dropout variance	Var(RUL) over 10 stochastic forward passes	≥ 0; > 0.05 → uncertain
Perturbation sensitivity	ΔRUL when feature *i* is zeroed, normalized by baseline	−1.0 to +1.0 per sensor
Data quality score	Weighted combination of distribution, uncertainty, drift	0.0–1.0; < 0.6 → FAULT
Wear multiplier (CNC)	Relative ECU-at-replacement across products, normalized to Product A	1.0× to ~2.5×

Uncertainty Propagation Through the Pipeline

The formulas above describe individual model outputs. The pipeline's behavior as a system depends on how uncertainty interacts across components.

The uncertainty character transformations identified in Part 1 have specific implementation consequences. At the sensor layer, uncertainty is additive: noise widens confidence bands but predictions remain in the right neighborhood. At the product classifier, uncertainty becomes a routing decision (Part 1): a misclassification selects a different wear trajectory entirely, producing errors proportional to the multiplier gap rather than to sensor noise. At the wear multiplier layer, the UC Berkeley-derived constants are embedded assumptions invisible to every confidence check the system performs, because they are parameters, not inputs.

Per-model confidence checks can therefore pass while the aggregate prediction carries a bias no individual checkpoint detects. The compound error is visible only at the output: when predicted RUL diverges from actual remaining life at replacement. This is why accumulated prediction error must be treated as a first-class input to current confidence. Each tool replacement narrows the structural uncertainty that per-model checks cannot see.

Scheduling Logic: OR-Tools CP-SAT

When a maintenance alert fires, the state engine passes the current job queue and the predicted RUL to the OR-Tools CP-SAT constraint solver. The solver formulates maintenance insertion as follows:

Variables: start time for each job and for the maintenance window, all integer-valued in minutes.

Hard constraints: No two tasks may run simultaneously on the same machine. All tasks must complete within the shift window. The maintenance window must complete before the current time plus the RUL deadline (RUL in cycles × tick interval in minutes).

Objective: minimize total schedule makespan (end time of the last task).

The solver finds the job sequence that inserts the maintenance window into natural gaps in the queue with minimum makespan increase. At robotic cell scale (8 jobs, one machine), CP-SAT proves optimality in under 2 seconds. The output is a before/after schedule pair: the baseline optimal schedule without maintenance, and the replanned schedule with maintenance inserted. The makespan delta quantifies the cost of the intervention.

The scheduler currently consumes the RUL point estimate and uses it as a hard deadline constraint. The RUL prediction is a distribution, not a point. A tool predicted at RUL = 50 cycles with ±10% accuracy could fail at cycle 45, and the maintenance window would arrive five cycles late. The scheduler's risk tolerance should be parameterized by the confidence interval rather than fixed to the point estimate, adjusted by 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.

Dynamic Scheduling: PPO Dispatcher

The CP-SAT solver produces a provably optimal schedule given a snapshot of the current state. It does not adapt as conditions change between replanning events. To extend scheduling into the Optimizing fidelity level defined in Part 1, the architecture includes a PPO (Proximal Policy Optimization) dispatcher that learns a dynamic dispatching policy from experience.

The dispatcher framework operates in a 77-dimensional state space encoding job queue, machine state, tool conditions, and RUL estimates, selecting among 6 dispatching rules including a HEALTH-priority rule. A policy has been trained on 5,000 synthetic job-shop scheduling instances. The framework is loaded at runtime, and its dispatching decisions are logged in the digital thread.

The PPO dispatcher's performance advantage over static dispatching has not yet been quantified against a baseline on the target cell's job mix. Until this validation is complete, CP-SAT remains the authoritative scheduling mechanism. The PPO framework represents the beginning of the Optimizing fidelity level: the infrastructure for learned, adaptive scheduling exists, the policy is trained, but the evidence that it outperforms the static solver on this specific workload has not been measured. This is an empirical question the deployment phase will answer, following the same principle that governs the Tier 3 sensor decision: the architecture supports the capability without requiring it.

Expected Outputs

Per tick: RUL in cycles, anomaly probability, health score, epistemic flag, data quality score, VSN attention weights (30 × 14 matrix), perturbation sensitivity scores (14 values).

Per alert: Alert record in maintenance log, OR-Tools before/after schedule pair with makespan values, Llama 3.2 plain-language advisory with top-5 sensor contributors, digital thread event.

Per operator action: Anomaly labels, maintenance records, job completion records, tool replacement records, shift snapshots, all written to the local state store and the digital thread.

Per dashboard tick (via SSE): All prediction fields, OODA loop phase, inference latency (P50, P95), streaming connection status.

Part 3.2: Dashboard and User Interface

3.2.1 Visualization and HCI Design Intent

The dashboard mockup is a running implementation served locally with no cloud dependency. Four stakeholder views share a single data stream; switching views never interrupts it.

Each view presents the same underlying data through the temporal lens that matches its audience's decisions: cycle for operators, shift for technicians, week for planners, quarter for managers.

Each panel answers a specific decision question rather than presenting data for interpretation.

Three confidence indicators persist across all views: model confidence meter, composite health score (0–100), and the four-state epistemic flag. These are primary visual elements, not metadata.

The entire dashboard is state-responsive. When the epistemic flag changes, every view adapts: advisory panels surface for operators, schedules shift to provisional for planners, relevant sensors are highlighted for technicians, and impact estimates update for managers. The four epistemic states produce four distinct dashboard configurations, each described in the stakeholder views that follow.

3.2.2 Actuation and Control Design Intent

Human-facing actuation. The Operator view includes a full operator input panel with five tabbed forms corresponding to the five operator input types defined in Part 1: maintenance confirmation, anomaly acknowledgment, job completion report, machine status override, and shift context. Each form captures structured data with the context needed for the feedback loops described in Part 3.1: tool ID, timestamp, technician ID, condition observed, parts used, and free-text notes. The Technician view adds two additional input workflows: optical scan entry (VBmax measurement with method selection and file upload for wear photographs) and tool management (remove, measure, or replace, with tool selection and replacement confirmation). These forms implement the ground-truth data paths described in Part 2.2.

When the epistemic state is Uncertain/Ambiguous, the Operator view presents a hypothesis panel with competing explanations and explicit tradeoff cards. In the demonstration scenario, two hypotheses (sensor fault at 52% vs. bearing degradation at 48%) are presented alongside three action options (run diagnostic, continue production, preemptive halt), each with a quantified cost and risk. This interaction pattern makes the system's uncertainty actionable rather than merely visible: the operator sees not just that the system is uncertain but what the options are, what each costs, and what each risks.

Machine-facing actuation. The OPC UA bidirectional capability described in Part 2.2 supports writing maintenance flags back to the SINUMERIK ONE HMI. In the current implementation, the system uses MQTT for demonstration. In production deployment, the edge agent would write to the controller's OPC UA node address space. The bidirectional capability means the twin can surface its maintenance recommendations directly on the machine's operator panel.

3.2.3 Stakeholder Views

Operator View (Cycle Timescale)

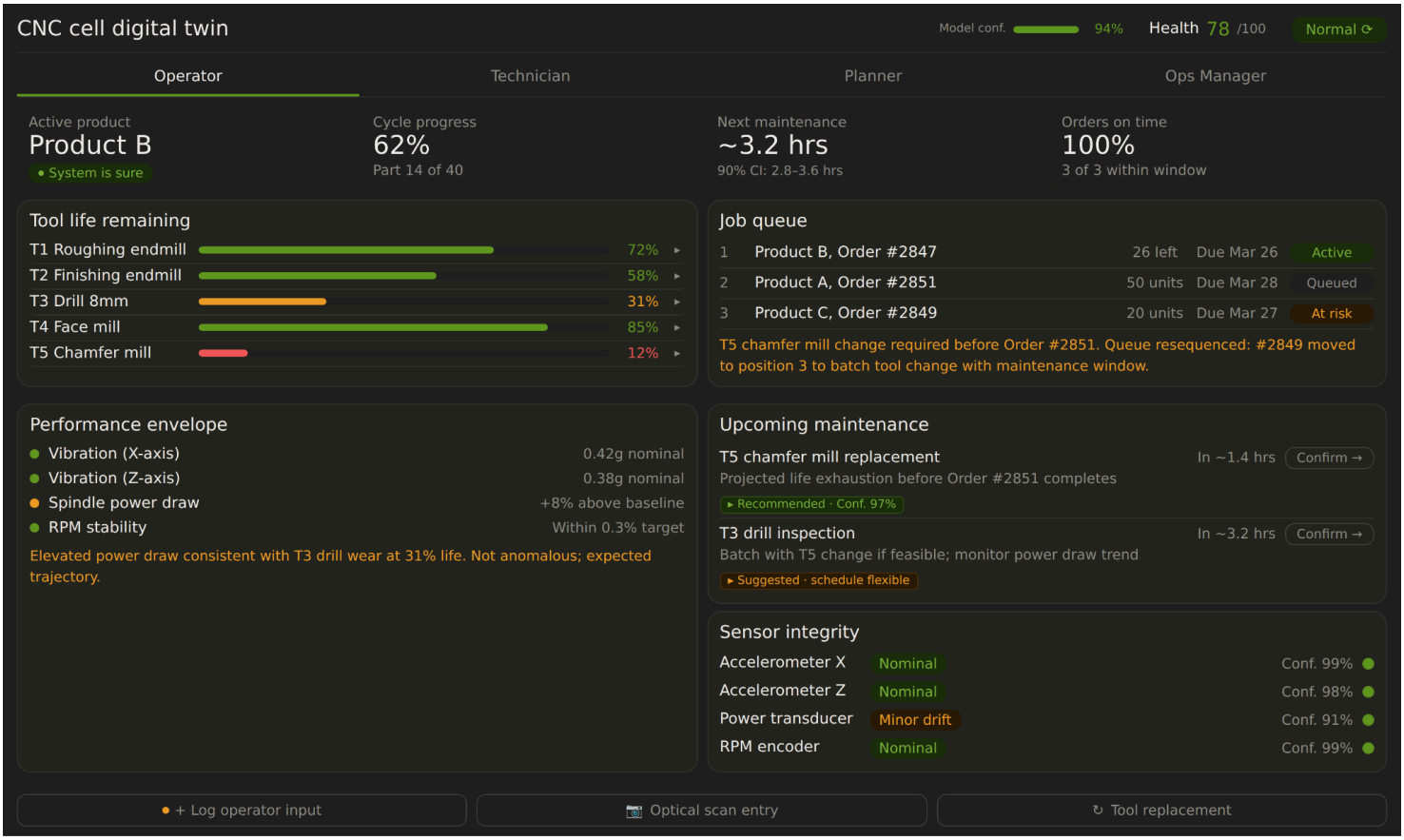


Figure 3.2a. Operator view, Normal state. Health 78, model confidence 94%. Tool life bars with click-to-expand drill-down, job queue with scheduler-driven resequencing, performance envelope, upcoming maintenance with confidence-tagged recommendations, and sensor integrity with calibration quality indicators.

The Operator view is the real-time face of the system. Its layout matches the reference dashboard concept introduced in Part 1 (Figure 1.5), with additions for health score, epistemic flag, and confidence intervals.

Four summary metrics span the top: active product (with classification confidence), cycle progress, next maintenance window, and orders on time. Below these, six panels are arranged in a two-column grid.

Tool life remaining. Per-tool bar chart showing remaining life as a percentage. Color shifts from green (> 40%) to amber (20–40%) to red (< 20%). Each tool row is interactive: clicking expands a detail panel showing the tool's RUL timeseries over the last 50 cycles with 90% and 50% confidence interval envelopes, the product mix that tool has cut (segmented bar), cumulative equivalent cutting units, and the most recent in-situ probing measurement (installed vs. current length and diameter, with the delta highlighted when it exceeds a wear threshold). Only one tool detail panel is open at a time.

Job queue. Active and queued orders with product type, quantity remaining, due date, and status badge (Active, Queued, At Risk, Halted). When the scheduler resequences the queue, an annotation below the list explains what changed and why.

Performance envelope. Per-sensor status indicators with current values. Status dots shift from green to amber when a reading is elevated but consistent with expected tool wear, and from amber to red when the reading is

anomalous. An annotation distinguishes between expected trajectories (elevated power draw at 31% tool life) and genuine anomalies.

Upcoming maintenance. Scheduled maintenance events with estimated time-to-window and the reasoning behind each recommendation. When the epistemic flag is not Normal, a warning annotation notes that timing estimates depend on a sensor currently flagged for review.

Sensor integrity. Per-sensor health with status badge (Nominal, Minor Drift, Drift Detected, Ambiguous), confidence percentage, and a calibration quality indicator (filled ring for well-calibrated, half-filled for reduced accuracy, quarter-filled for low accuracy). This panel is the operator's view of the sensor integrity module described in Part 3.1.

Three action buttons at the bottom provide shortcuts: Log Operator Input (opens the five-tab input panel), Optical Scan Entry (navigates to the Technician view's scan form), and Tool Replacement (navigates to the Technician view's tool management workflow).

When the epistemic state shifts to Uncertain/Ambiguous, the panels above are overlaid by a hypothesis panel presenting the competing explanations, their probabilities, and three option cards with explicit tradeoffs. This replaces the standard advisory banner used in the Sensor Drift and Degradation states.

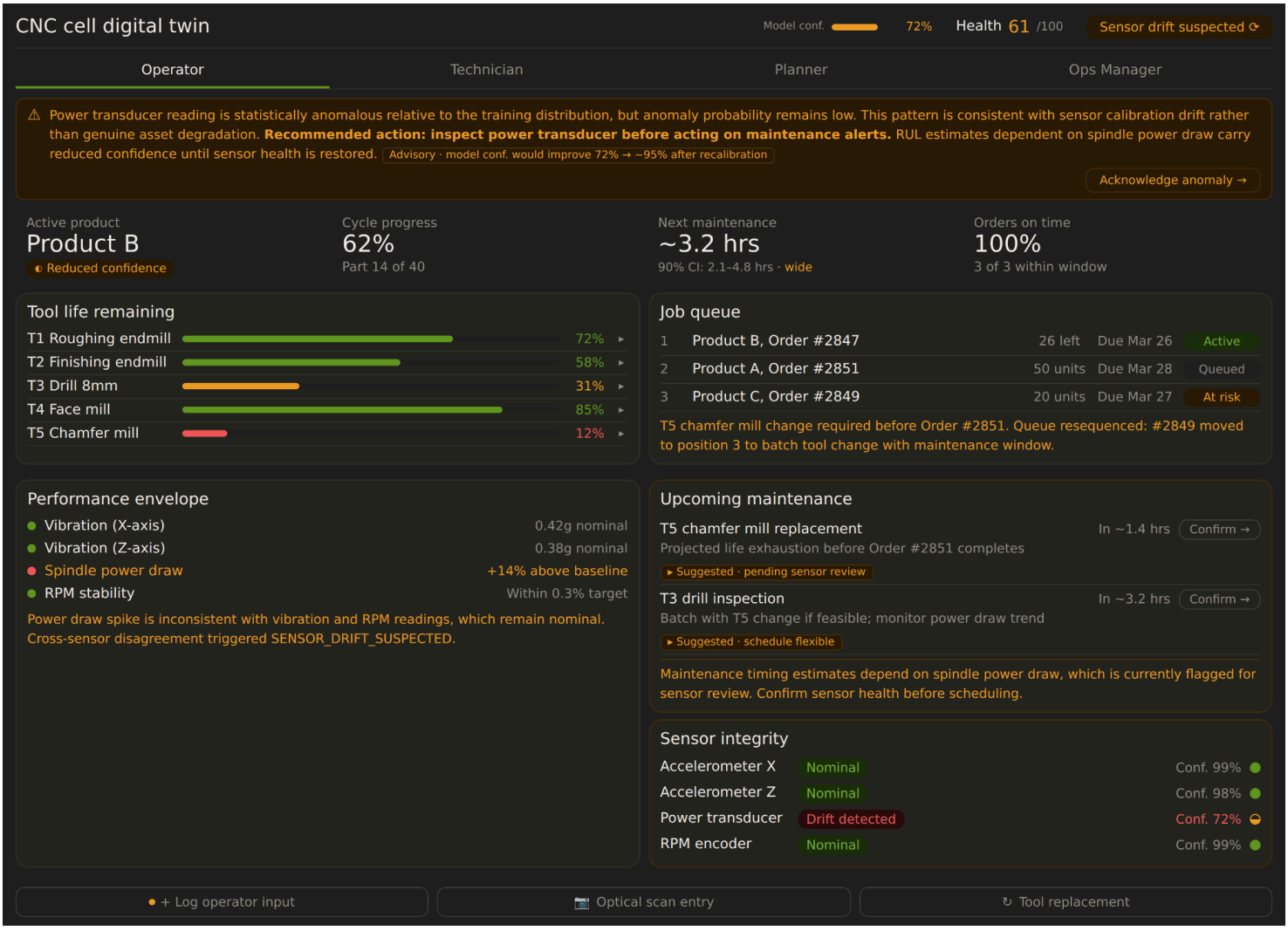


Figure 3.2b. Operator view, Sensor Drift Suspected state. Advisory banner recommends sensor inspection before acting on maintenance alerts. Product classification confidence drops from 97% to 84%. Maintenance timing shifts to "pending sensor review."

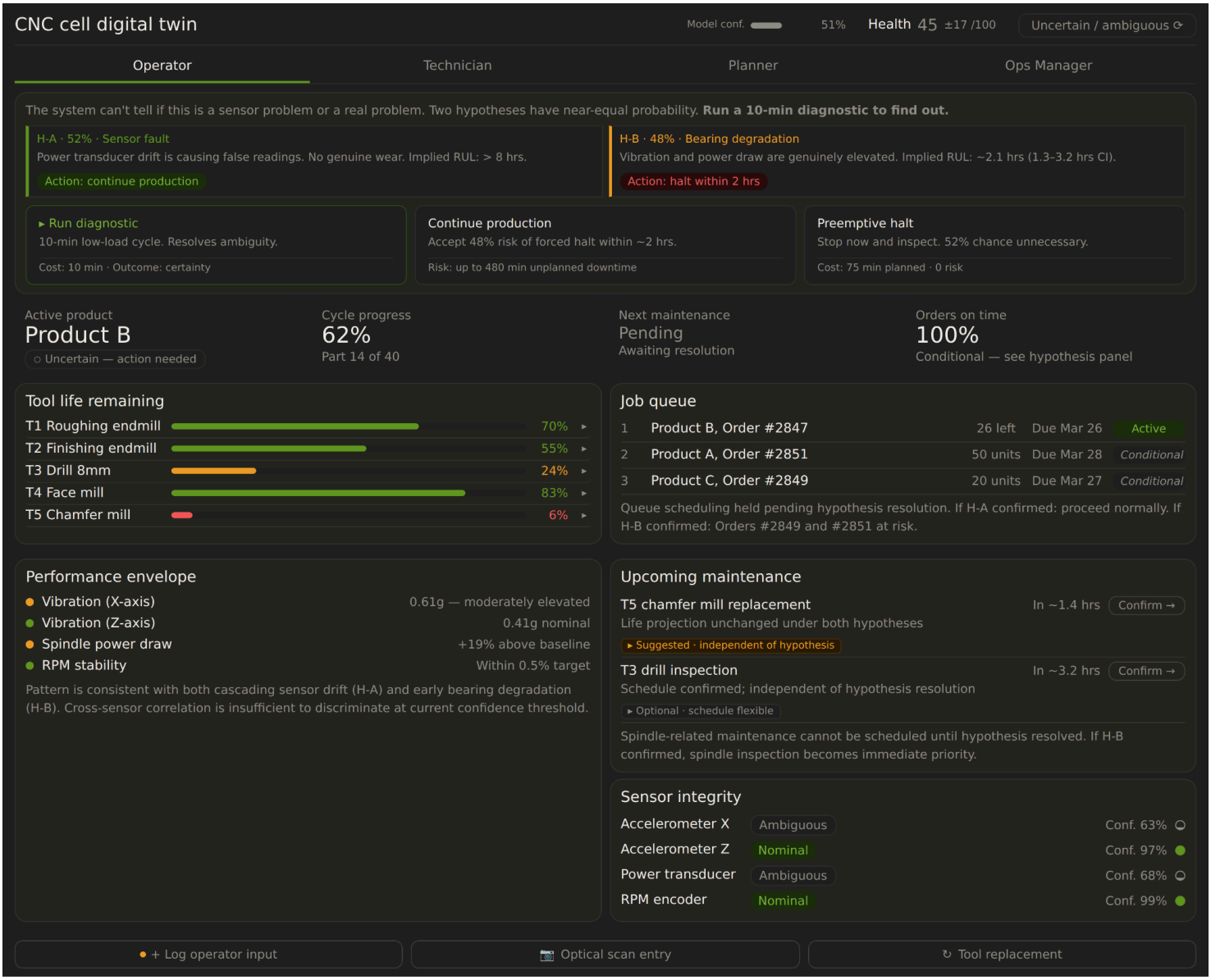


Figure 3.2c. Operator view, Uncertain/Ambiguous state. Hypothesis panel presents two competing explanations (52% sensor fault vs. 48% bearing degradation) with three action options. Each option card quantifies its cost and risk.

Technician View (Shift Timescale)

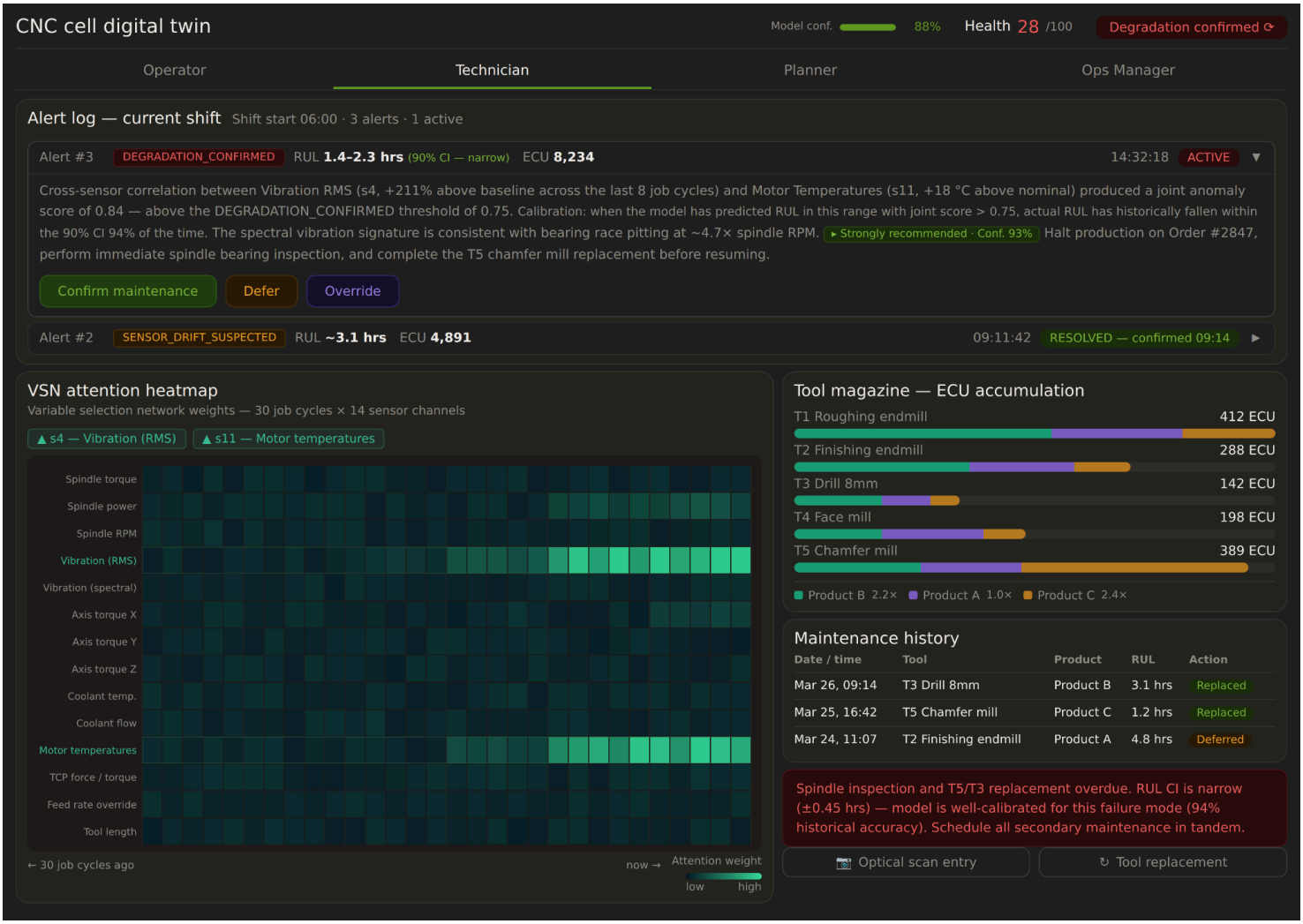


Figure 3.2d. Technician view, Degradation Confirmed state. Expanded alert shows cross-sensor correlation diagnostic with confidence-tagged recommendation. VSN attention heatmap highlights sensors s4 and s11. Tool magazine shows per-product ECU accumulation with wear multipliers. Maintenance history table and ground-truth data entry buttons at bottom.

The Technician view shifts temporal resolution from the current cycle to the current shift. It contains three primary sections plus two sub-panel workflows.

Alert log. Alerts from the current shift as expandable accordion rows. Each row shows the alert number, RUL at alert time, epistemic flag, and timestamp. Expanding a row reveals the Llama-generated diagnostic report (which sensors drove the alert, what action is recommended), the severity classification, and action buttons (confirm maintenance, defer, override). Confirmed and resolved alerts collapse to a single summary line. The technician's decision, logged with its full context, becomes training data for the feedback loops.

VSN heatmap. The Variable Selection Network's attention weights rendered as a 30-cycle × 14-sensor color grid. Most cells are dim during normal operation; concentrated brightness on specific sensors in recent cycles indicates the model has identified a degradation signal. Highlight chips above the heatmap call out which sensors the model is attending to most strongly.

Tool magazine. Per-tool wear accumulation as stacked horizontal bars segmented by product type (Product A, B, C), showing how each tool's wear distributes across the product mix. The wear multiplier values (A = 1.0×, B =

2.189×, C = 2.406×) are displayed alongside the legend. Clicking a tool row in the magazine navigates to the tool management sub-panel.

Optical scan entry (sub-panel). A dedicated form for logging VBmax flank wear measurements from offline 3D profilometry or optical inspection. Fields include tool ID, measurement timestamp, VBmax value, measurement method, wear photograph upload (PNG, JPG, PDF, STL), and free-text notes for wear location, chipping, and observations. On submission, the scan data is written to the digital twin event log and incorporated in the next inference cycle.

Tool management (sub-panel). A workflow for removing, measuring, or replacing end-mill tools. The technician selects a tool from a visual grid showing current life percentage, then selects an action (remove, measure, replace). Replacement resets RUL to 100% and ECU to zero, creating the ground-truth data point that the recalibration loop consumes. Each action produces a structured record in the maintenance log.

Planner View (Weekly Timescale)

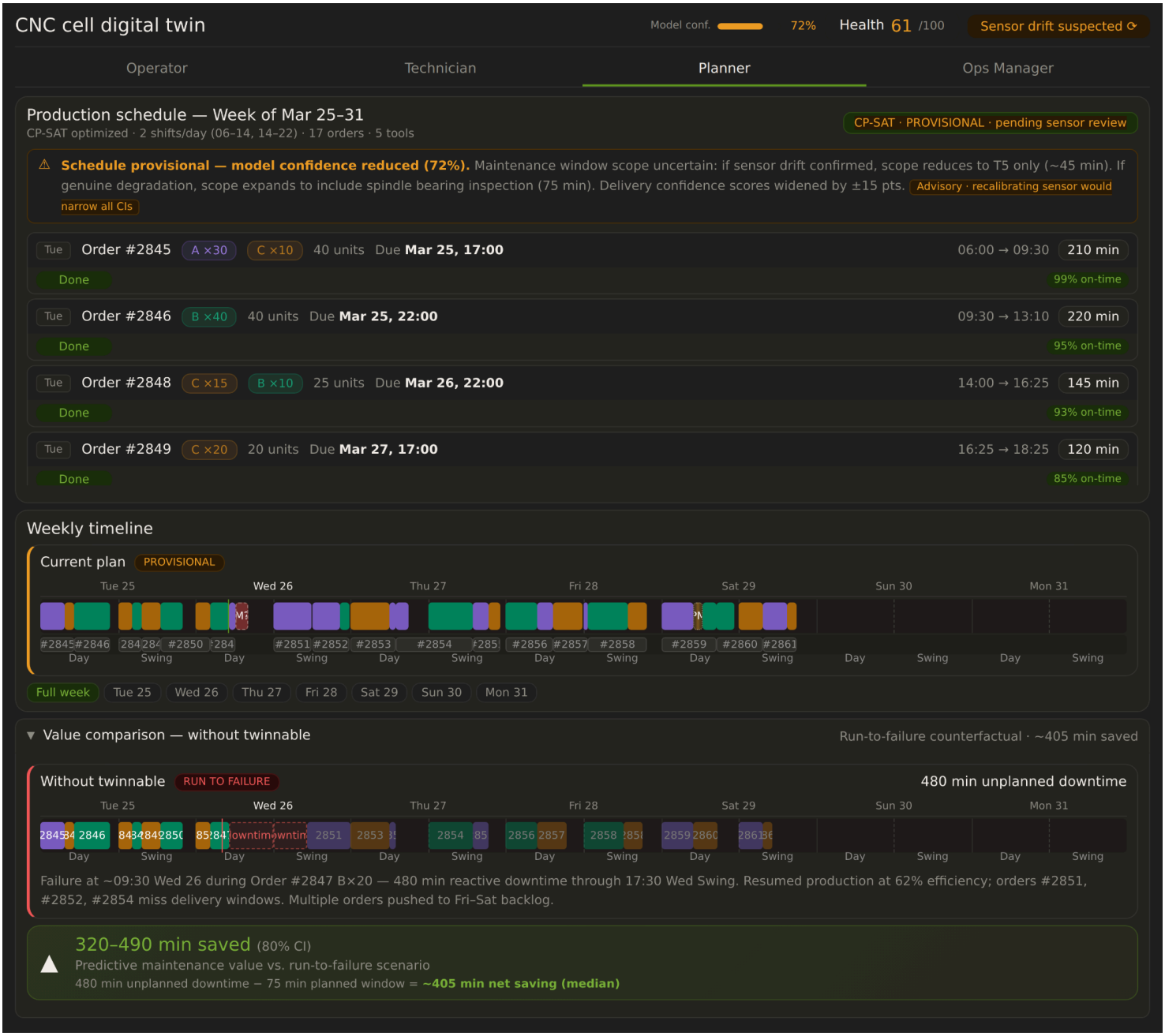


Figure 3.2e. Planner view, Sensor Drift Suspected state. Schedule marked PROVISIONAL with pending sensor review. Delivery confidence scores widened. Value comparison accordion open, showing run-to-failure counterfactual (480 min unplanned downtime) and predictive value (320–490 min saved, 80% CI).

The Planner view presents the production schedule for the current week (7 days × 2 shifts per day) as a horizontal timeline Gantt chart. Job bars are color-coded by product type and proportionally sized by duration. A "now" indicator marks the current position in the schedule. Day-level zoom is available for inspecting individual shifts, with shift-change markers and hourly gridlines at the zoomed level.

The view is state-responsive across all four epistemic states:

Normal: The CP-SAT optimized schedule is displayed with all maintenance windows placed. Order cards below the Gantt show each order's status, product runs, due date, and a delivery confidence chip (High, Medium, At

Risk). Clicking a job bar in the Gantt opens a detail card showing order metadata, required tools with their current and projected post-job life percentages, and estimated duration.

Sensor Drift Suspected: The schedule shifts to provisional. A pending maintenance window is shown with a dashed border, indicating it is held until sensor health is confirmed. The resequencing annotation explains that the maintenance timing depends on a sensor currently under review.

Degradation Confirmed: The schedule shows the replanned configuration with the maintenance window inserted. The resequencing annotation quantifies the makespan impact.

Uncertain/Ambiguous: The schedule is held. Conditional job runs are dimmed. The annotation explains that the schedule cannot be optimized until the ambiguity is resolved.

A collapsible "Value comparison" section presents the run-to-failure counterfactual: the three-panel comparison (without twinnable / baseline / replanned) with the predictive value callout (~405 minutes saved per failure event) and the alert metadata footer (trigger RUL, solver status, maintenance window duration, avoidable disruption).

Ops Manager View (Quarterly Timescale)

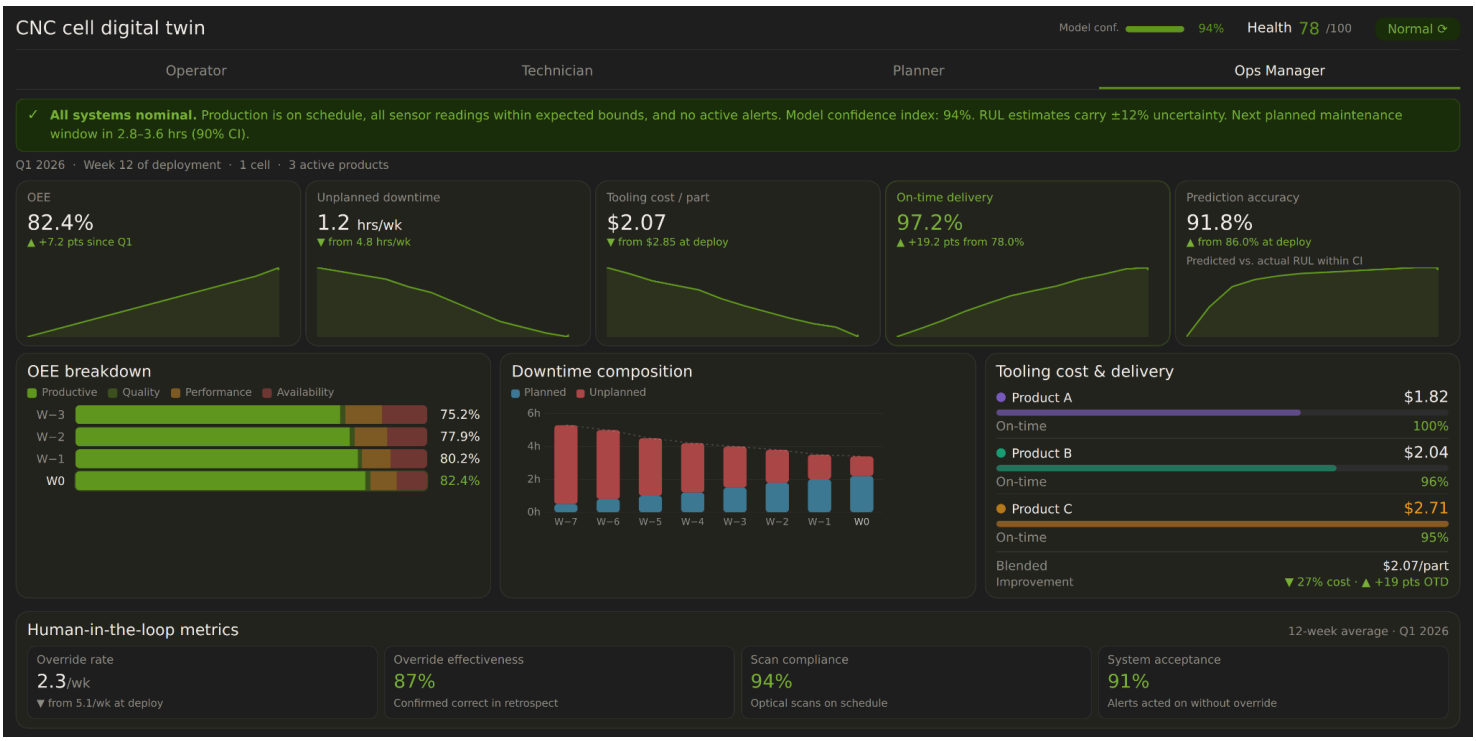


Figure 3.2f. Ops Manager view, Normal state. Five summary metrics with sparkline trends (OEE 82.4%, unplanned downtime 1.2 hrs/wk, tooling cost $2.07/part, on-time delivery 97.2%, prediction accuracy 91.8%). OEE breakdown, downtime composition, per-product cost and delivery, and human-in-the-loop co-adaptation metrics.

The Ops Manager view addresses the management stakeholder identified in Part 1's stakeholder matrix. It operates at the quarterly timescale and presents five summary metrics with expandable detail charts.

Summary metrics (top row): OEE (82.4%, trending up 7.2 points since Q1 start), unplanned downtime (1.2 hrs/week, down from 4.8), tooling cost per part ($2.07, down from $2.85 at deployment), on-time delivery (97.2%, up from 78.0%), and prediction accuracy (91.8% of predictions within confidence interval). Each metric card includes a sparkline showing the 12-week trend. Clicking a card expands a detail panel with the corresponding chart (OEE decomposition, downtime trend, cost convergence, or accuracy convergence).

OEE breakdown. Stacked horizontal bars for the last four weeks, decomposed into productive time, quality loss, performance loss, and availability loss. The improving trend from 75.2% to 82.4% is visible in the growing productive segment and shrinking availability loss.

Downtime composition. Planned vs. unplanned downtime per week. The story these bars tell: unplanned downtime decreases as the system learns, while planned downtime increases slightly (more proactive interventions). Net downtime decreases.

Tooling cost and delivery. Per-product cost and on-time delivery rate. Product C ($2.71/part, 2.4× multiplier) is the most expensive; Product A ($1.82/part, 1.0× multiplier) is the least. Blended cost improvement of 27% alongside a 19-point on-time delivery improvement quantifies the system's operational impact.

Human-in-the-loop metrics. Four indicators that track the co-adaptation between operators and the system: override rate (2.3/week, down from 5.1), override effectiveness (87% confirmed correct in retrospect), scan compliance (94% of scheduled optical scans completed), and system acceptance (91% of alerts acted on without override). These metrics are the operational measurement of the trust calibration described in Part 3.1's feedback architecture.

A state-responsive banner at the top of the Ops Manager view provides a management-level summary of the current system state, quantified in terms that matter at this timescale: model confidence index, RUL uncertainty range, number of orders at risk, and estimated current-shift impact of the recommended action. The banner includes a confidence-tagged recommendation (e.g., "Strongly recommended, Conf. 93%") so that the manager sees not just what the system recommends but how confident it is in that recommendation.

Thread and Graph Views (Engineering/Audit)

The running twinnable implementation includes two additional views for engineering and audit purposes, accessible through the backend API and the Svelte dashboard but not included in the operator-facing mockup.

The *Thread view* provides the digital thread timeline and provenance graph. The timeline lists every meaningful state transition (sensor reading → prediction → alert → operator decision) with OODA phase annotations. The provenance graph renders these as a D3 force-directed graph with causal edges. A state-change detection filter reduces a typical shift to 5–12 meaningful nodes. These views serve quality auditors and system engineers who need to reconstruct the causal chain behind any maintenance decision.

The *Graph view* shows the knowledge graph: machines, tools, products, and sensors as typed nodes; *contains*, *monitors*, and *produces* as typed edges. Selecting a node reveals its observation chain. This view serves the engineering audience that needs to understand structural relationships rather than temporal events.

3.2.5 Technical Platforms and Tools

The implementation runs entirely on the local machine with no cloud dependency.

Frontend: Svelte 5, Vite, TypeScript. D3.js for the Twin State Timeline, VSN Heatmap, Gantt charts, and knowledge graph. Canvas API for sparklines and RUL timeseries with confidence interval envelopes. SSE via the native browser EventSource API.

Backend: Python 3.11, FastAPI, Uvicorn. ONNX Runtime for inference. PyTorch for the explanation path. OR-Tools for CP-SAT scheduling. Stable-Baselines3 for PPO policy loading.

Storage: SQLite for operational state, maintenance log, digital thread, and feedback tables. TimescaleDB via Docker for the live sensor stream (90-day retention, hypertable auto-compression). SQLite fallback in demonstration mode.

Local LLM: Llama 3.2 via Ollama. No API key, no cloud dependency. Falls back to a structured text template if Ollama is not running.

Testing: 238 backend tests (pytest), 154 frontend tests (Vitest), 8-step automated demo walkthrough (Playwright, exit code 0).

The system deliberately avoids cloud-dependent or proprietary visualization platforms. The entire stack runs on a laptop with no internet connection after initial setup, a design choice for deployability in industrial environments where network access to production equipment may be restricted.

Part 3.3: Analytics and Value Proposition Summary

3.3.1 Analytics and Decision Support

The digital twin supports nine analytic tasks. Four are primary decision drivers; five are supporting capabilities that make the primary tasks trustworthy.

Primary analytic tasks:

RUL regression converts a 30-cycle sensor window into a remaining useful life estimate. TwinnableTFT achieves $R^2 = 0.889$ on CMAPSS; the NUAA Stack-LSTM achieves $R^2 = 0.9992$ on CNC titanium milling (W1 condition), exceeding published benchmarks.

Product classification identifies which of three aluminum products is being machined (99.93% validation accuracy on TUAWS), enabling automatic wear multiplier selection without manual changeover logging.

Maintenance schedule optimization inserts a maintenance window into the production queue with minimum makespan disruption. CP-SAT proves optimality in under 2 seconds. The standard demonstration shows a net saving of approximately 405 minutes per failure event compared to unplanned breakdown.

Epistemic classification distinguishes sensor hardware failures from genuine asset degradation using the four-state flag described in Parts 1 and 3.1.

Supporting analytic tasks:

Feature attribution identifies which sensors drove a given RUL estimate, using perturbation sensitivity validated against independent SHAP importance rankings ($\rho = 0.802$).

Anomaly detection identifies deviations from nominal behavior before they manifest as RUL degradation (NIST anomaly head AUC = 0.9276).

CNC-specific tool wear RUL predicts remaining tool life in cutting seconds from 300 Hz milling signals.

Diagnostic report generation converts statistical model output into actionable plain-language maintenance instructions via Llama 3.2.

Digital thread provenance creates a verifiable audit trail from sensor reading to maintenance action, supporting regulatory compliance, warranty disputes, and retrospective failure analysis.

Analytics Summary Table

Analytic Task	Data Inputs	Analytical Method	Expected Outcome / Decision Impact
RUL regression	30-cycle sensor window (14 channels), condition cluster	TwinnableTFT (ONNX); CMAPSS $R^2 = 0.889$	Quantified maintenance horizon; alert fires at RUL ≤ 30
Product classification	Current sensor window, cutting signal patterns	Sequence classifier (TUAWS); 99.93% accuracy	Selects correct wear multiplier; prevents ×2.4 RUL error
Anomaly detection	Sensor window, training distribution statistics	Binary classification head + Mahalanobis distance; AUC = 0.9276	Early deviation detection before RUL impact
Epistemic classification	Mahalanobis distance, MC dropout variance, drift score	Three-signal rule-based classifier → four-state flag	Separates sensor fault from degradation; prevents wrong-response errors
Feature attribution	Inference window, model checkpoint	Perturbation sensitivity; VSN-SHAP ρ = 0.802	Directs technician to the right subsystem
CNC tool wear RUL	300 Hz milling signals (8 channels), WPT features	Stack-LSTM (3 layers); W1 $R^2 = 0.9992$, W2 $R^2 = 0.9945$	Per-tool RUL in cutting seconds
Schedule optimization	Job queue, shift window, RUL deadline	OR-Tools CP-SAT; proves optimality in < 2 s	Inserts maintenance with minimum production cost
Diagnostic report	Top-5 perturbation scores, RUL, domain context	Llama 3.2 (local); one report per alert	Plain-language maintenance instructions
Digital thread provenance	All state transitions, operator decisions, replacements	State-change detection filter → causal event graph	Verifiable audit trail from sensor to action

3.3.2 Insight, Outcomes, and Success Evaluation

The value proposition defined in Part 1, converting unplanned maintenance into scheduled maintenance while surfacing the system's own confidence, is quantifiable in the implementation. The Schedule view's three-panel Gantt comparison shows the difference: unplanned failure produces roughly 1,035 minutes of makespan; planned maintenance produces 630 minutes. The net saving is approximately 405 minutes per failure event. In a CNC cell running a 40-week production year with historically one unplanned failure per quarter, this represents approximately 36 hours of recovered production time annually, before accounting for scrap, rework, rush order premiums, and tool damage.

The system's value strengthens over time as the feedback channels described in Part 3.1 replace assumed confidence with measured confidence from this specific cell's operational history. 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 estimates, 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 system gets more honest about its own accuracy the longer it runs, and its honesty compounds.

Success Criteria

Criterion	Target	Achieved
Prediction accuracy (CMAPSS)	RMSE ≤ 15 cycles, $R^2 \geq 0.85$	RMSE = 12.06, $R^2 = 0.889$ ✔
Prediction accuracy (NUAA CNC)	$R^2 \geq 0.96$ (published benchmark)	W1 $R^2 = 0.9992$, W2 $R^2 = 0.9945$ ✔
Epistemic attribution correlation	$\rho \geq 0.75$ vs independent ranking	$\rho = 0.802$ ✔
Scheduling optimality	CP-SAT OPTIMAL in ≤ 5 s	OPTIMAL in < 2 s ✔
Anomaly detection (NIST)	AUC ≥ 0.90	AUC = 0.9276 ✔
Product classification (TUAWS)	Accuracy ≥ 98%	99.93% ✔
False alarm reduction	SENSOR_DRIFT correctly separates sensor from degradation	Validated by design; longitudinal field validation pending
System reliability	All tests passing	238 backend + 154 frontend + 8/8 Playwright ✔

Justifications. *RMSE ≤ 15 cycles:* 15 cycles represents roughly the minimum actionable lead time for inserting a maintenance window into a typical shift's queue. *AUC ≥ 0.90:* conventional threshold for anomaly detectors where false positives and false negatives carry comparable operational cost. *Accuracy ≥ 98%:* each misclassification selects the wrong wear multiplier, introducing an error bounded by the multiplier gap; 98% caps expected routing error at roughly one miscoded cycle per 50. *False alarm rate < 20%:* set by operator attention economics rather than statistical argument; above roughly one in five, operators tend to begin dismissing alerts without reading them. *CP-SAT ≤ 5 s:* bounded by the tick interval, since a solver slower than a tick produces a schedule for a state that has already changed. These targets are calibrated to the cell's current operating envelope: its shift structure, queue density, product mix, and staffing model. If the envelope changes, the targets should be revisited. An RMSE of 15 cycles assumes a specific relationship between cycle count and the lead time needed to insert a maintenance window. A false alarm threshold of 20% assumes a specific operator attention economy. A cell running three shifts with dedicated monitoring staff

tolerates a different false alarm rate than a cell where one operator manages alerts alongside active machining. The targets are not arbitrary, but they are contextual.

Section 4: Governance, Validation, and Communication

Part 4.1: Maintenance and Validation Plan

Sustaining Accuracy Across Operational Time

The digital twin's predictions at deployment rest on three categories of assumption, each degrading on a different timescale, through a different mechanism, and requiring a different validation response.

Assumption Category	Degradation Mechanism	Timescale	Detection Method
Sensor calibration: physical signals reflect the phenomena they claim to represent	Mounting loosening, cable fatigue, thermal drift, fouling	Weeks to months	Cross-sensor consistency checks (Part 3.1)
Model calibration: learned parameters generalize from training distributions to this cell	Distribution shift as product mix, tooling, materials, or ambient conditions evolve	Months to quarters	Mahalanobis distance and distribution drift signals in the epistemic integrity layer (Part 3.1)
Structural calibration: wear multipliers, alert thresholds, and envelopes match this cell's behavior	Accumulated operational evidence reveals divergence between prior values and asset-specific reality	Months to first recalibration	Rolling prediction error at tool replacement events

The validation lifecycle operates at three nested timescales: the 30-second tick cycle that governs real-time inference, the replacement-event cycle that governs parameter recalibration, and the quarterly-to-annual cycle that governs structural review.

Update Frequency and Data Refresh

The state engine's 30-second tick cycle is the system's heartbeat. At each tick, the engine queries the most recent 30 sensor readings, constructs the input tensor, runs inference, evaluates alert conditions, updates the state store, pushes results to dashboard clients via server-sent events, and writes a causal event to the digital thread.

Three mechanisms detect data quality issues before predictions reach any stakeholder. Cold-start suppression withholds all predictions during the first 30 cycles after a reset. Late data handling filters records older than a configurable latency threshold. The epistemic integrity layer gates every prediction with a four-state flag derived from three signals, as described in Part 3.1. Alert thresholds and production rate parameters are hot-reloadable without server restart. Model weights require a staged reload that swaps weights atomically mid-operation.

Recalibration Triggers

Per-product wear multipliers recalibrate after a minimum of three tool replacement events per product type. The recalibration endpoint replaces the initial priors with multipliers computed from actual equivalent cutting units at replacement on this specific machine. This is the slowest feedback channel and the one that determines the system's overall convergence rate. Two metrics track convergence: the rolling coefficient of variation across the

three most recent multiplier estimates per product, where declining CV indicates convergence and stalling CV indicates stabilization or genuine variability; and the rolling prediction error at each replacement event. If prediction error fails to decline over a window of ten replacement events, the system flags a convergence stall and recommends structural revision rather than continued parameter tuning.

Condition scaler drift triggers recalibration when the epistemic flag transitions to SENSOR_DRIFT_SUSPECTED persistently without escalating to DEGRADATION_CONFIRMED. This pattern is the canonical signature of sensor hardware drift rather than genuine asset degradation. The recommended response is sensor inspection, not maintenance scheduling. The validation plan makes operational the boundary between graceful degradation and epistemic failure that Part 1 identified architecturally.

Domain head retraining triggers after 50 or more labeled anomaly events per domain accumulate through operator feedback. The fine-tuned head is validated against the same acceptance criteria as the original before deployment.

Performance Monitoring

KPI	Target	Monitoring Mechanism
RUL prediction accuracy	RMSE ≤ 15 cycles (CMAPSS); nRMSE ≤ 0.30 (all domains)	Predicted-vs-actual comparison at each tool replacement event
Inference latency	< 100 ms per tick	P50 and P95 exposed on dashboard; ONNX Runtime CPU inference
Epistemic flag distribution	SENSOR_DRIFT_SUSPECTED < 10% of ticks	Per-tick logging to twin state; persistent drift triggers maintenance review
Product classification accuracy	Tracked against 99.93% validation benchmark	Rolling accuracy from operator verification at changeover
False alarm rate	< 20% per rolling window	Ratio of operator-marked false alarms to total alerts

These per-model metrics cannot detect compound error: the case where individual confidence checks pass while the aggregate prediction carries a bias no single checkpoint can see. Compound error emerges at model interfaces where uncertainty changes character. Sensor noise enters the pipeline as additive variance, widening the confidence band around a prediction. When that noisy signal crosses into the product classifier, uncertainty transforms from a continuous variance into a discrete routing decision: which product regime, and therefore which wear multiplier, governs the RUL estimate. If the classification is wrong, the downstream error is proportional to the difference between the selected regime and the correct one, not to the input noise that caused the misclassification. Per-model confidence checks can each pass while the aggregate prediction carries a bias that no individual checkpoint is positioned to detect. The only measurement that catches this class of error is the prediction-versus-actual comparison at tool replacement. Each replacement event is treated as a first-class validation opportunity, and the cross-model validation it provides is the reason the recalibration architecture treats replacement events as the system's primary source of ground truth.

Error Detection, Escalation, and Recovery

Escalation follows the four epistemic states defined in Part 3.1 and visualized across the dashboard configurations in Part 3.2. The governance contribution is naming the autonomy boundary at each state. NORMAL permits autonomous operation within defined bounds. SENSOR_DRIFT_SUSPECTED directs

operators to inspect sensor hardware rather than schedule maintenance. DEGRADATION_CONFIRMED halts production and presents full diagnostic context. UNCERTAIN_AMBIGUOUS withholds scheduling decisions entirely, presenting competing hypotheses with quantified tradeoffs until operator resolution.

Recovery is automatic where possible. The state engine reinitializes from configuration and local storage on restart. Cold-start suppression resumes inference once 30 cycles are available. If the local Llama instance is unreachable, a structured text template substitutes for the narrative advisory without interrupting the alert or scheduling pipeline. If model weights fail integrity checks, the system exposes the discrepancy through the health endpoint.

Risk Prioritization

Risk	Impact	Timescale	Detection / Mitigation
Epistemic failure: system acting confidently on compromised data	Highest	Immediate (cycle-level)	Sensor integrity module, cross-sensor consistency, four-state flag. Structural mitigation: architecture makes this detectable or forces widened uncertainty.
Censored data: conservative operator behavior biases training toward early intervention	Highest	Months to years	Survival analysis (Kaplan-Meier, Cox proportional hazards) handles censored observations natively. Monitor replacement-vs-predicted RUL gap distribution for persistent skew.
Product classifier degradation from material property shift	Moderate	Months	Rolling classification accuracy tracked against operator verification at changeover.
Wear multiplier instability for variable machining conditions	Moderate	Months	Rolling CV of multiplier estimates; convergence stall detection.
Synthetic-to-real generalization gap (robot joint model)	Moderate	At deployment	ESTIMATED badge policy; epistemic flag distinguishes model uncertainty from sensor issues.

Lifecycle Planning

Maintenance at the shortest timescale is the tick cycle; maintenance at the longest timescale is the lifecycle of the twin itself. Four planned cycles govern long-term operation, each addressing a different class of drift between the twin and the cell it models.

Cycle	Trigger	Scope	Owner
Calibration	Accumulated replacement events (≥3 per product) or persistent sensor drift flag	Wear multipliers, condition scalers, alert thresholds	Model owner with maintenance technician input
Sensor Replacement	End-of-service-life on individual sensors, or cumulative calibration drift beyond recovery	Physical sensor swap with continuity-of-record preservation in the digital thread	Data steward
Technology Refresh	Dependency end-of-support (inference runtime, database, dashboard framework) or capability gaps identified in requirements review	Infrastructure components, without changes to model architecture or trained weights	Operations liaison with system engineering
Requirements Review	Annual, or triggered by significant operational change (new product line, new cell)	Whether the twin's current capabilities still match the operational decisions it is asked to support	Model owner with operations management

The calibration cycle is described in detail above. The sensor replacement cycle requires specific care for continuity of the historical record. When a sensor is replaced, its successor must be cross-calibrated against overlapping measurements where possible, and the digital thread must record the replacement as a named

event so that downstream analysis can reason about signal characteristics on each side of the boundary. Without this continuity, a sensor swap appears to the wear model as a sudden distribution shift, triggering spurious SENSOR_DRIFT_SUSPECTED flags and potentially corrupting the multiplier recalibration path.

Recalibrate, Upgrade, or Rebuild

When fidelity loss is detected, three intervention paths are available, ordered by cost and structural disruption.

Recalibrate when the twin's structure is sound but its parameter values no longer reflect the physical system. This is the appropriate response to sensor drift, distribution shift, or accumulated operational evidence that diverges from prior values. Recalibration is the cheapest intervention and should be the first option considered. The recalibration triggers described earlier in Part 4.1 operationalize this path.

Upgrade when the architecture remains viable but specific components require modification: a new domain head, a new sensor tier, an improved classifier, an expanded feedback channel. The federated twin architecture supports targeted upgrades by design: a tool twin can be upgraded without touching the cell twin, and a new domain head can be added without modifying the frozen backbone. Upgrades require integration testing to confirm that new components interact correctly with existing ones, and they must be managed through the version control and governance processes described in Part 4.2.

Rebuild when accumulated technical debt, architectural mismatch with current operational needs, or fundamental violation of original design assumptions makes targeted upgrades costlier than starting fresh. Rebuilding is a normal stage in the lifecycle of any long-lived technical system. The organizational knowledge accumulated during the first generation's operation, including where model assumptions broke down, which sensors proved most reliable, which use cases generated the most operational value, and what the governance structure actually needs to look like in practice, is the best possible specification for the successor system. A well-documented first-generation twin is the starting point for the second generation. What distinguishes a rebuild from a failure is precisely this: the first system produced the knowledge that the second system requires.

Retirement

Retirement is the final stage of a well-managed lifecycle. Responsible retirement involves four steps: documenting the institutional knowledge embedded in the twin's model and governance history so it is available to the successor system or to the operations team that will run the cell without a twin; preserving the historical data in a form that can be accessed without the twin's inference infrastructure, typically as exported TimescaleDB and digital thread archives in open formats; communicating clearly to the twin's users what capabilities are being retired and what, if anything, replaces them; and reviewing what was learned during the system's operational life that should inform the design of its successor. The retirement review produces the same kind of artifact as the requirements review described above, but at a different scale: the requirements review asks whether the current system still matches its use case. The retirement review asks what the next system should be.

Part 4.2: Data and AI Governance

The Digital Thread as Chain of Custody

Every state transition in the digital twin is traceable through a causal audit trail where each event references the events that produced it and the events it produced. The thread's retention policy preserves permanent events (alerts, operator decisions, tool replacements, model updates) at full fidelity indefinitely. Sensor data follows a three-tier retention architecture: full fidelity for 0 to 30 days, 5-minute aggregates for 30 to 90 days, hourly aggregates beyond 90 days. A quality auditor reconstructing the causal chain behind a maintenance decision six months after the fact will find every alert, every operator response, and the model state that produced them, linked to aggregated sensor context.

Data Provenance and Quality Standards

Data quality is enforced at three stages. At ingestion, shape and finiteness validation catches malformed input, with non-finite values triggering SENSOR_DRIFT_SUSPECTED rather than producing false predictions. At feature engineering, feature selection rationale is documented per domain in model metadata. At normalization, domain-specific condition scalers are maintained as versioned artifacts with runtime version-match verification. The dual normalization strategy described in Part 2.2, where the classifier and wear model receive differently transformed versions of the same raw input, means each model has its own data lineage, scaler version, and provenance record.

Model Governance and Version Control

Every trained model artifact carries a quality badge before deployment: VALIDATED meets all acceptance criteria on held-out test data; ESTIMATED is based on proxy metrics or datasets without published benchmarks, with reported accuracy treated as an upper-bound estimate; CALIBRATION denotes empirical parameters derived from operational data, such as wear multipliers. The frozen backbone principle documented in Part 3.1, where the TwinnableTFT backbone is never modified when adding domain heads, is an explicit governance constraint that protects the explainability of every domain the system serves. Model artifacts are stored with commit hashes logged in versioned metadata; the configuration version match check verifies weight integrity at runtime.

Accountability Chain

Link	Question	Held by	Governance Mechanism
Data quality	Is the training data complete, representative, and free from historical bias?	Data steward (per twin scope)	Feature engineering provenance, dual normalization strategy, sensor health monitoring
Validation	Is the model's performance verified against independent test data before deployment?	Model owner	VALIDATED / ESTIMATED / CALIBRATION badge system; acceptance criteria per domain
Thresholds	Who defines the alert levels that translate model outputs into recommended actions?	Operations management with operator feedback	Hot-reloadable configuration; accumulated feedback enables asset-specific adaptation

Monitoring	Who tracks performance over time and triggers retraining or replacement?	Model owner	Recalibration triggers, convergence stall detection, KPI monitoring
Decision Ownership	Who bears responsibility for decisions made on the model's outputs?	Operator (in degraded states); system (in NORMAL state, within defined bounds)	OODA autonomy boundary tied to epistemic state; digital thread records decision authority at each step

Transparency at Three Levels

Explainability operates at three levels, each answering a different stakeholder question.

Model level. Which input features does the model weight most heavily, and do those weightings make physical sense? The Variable Selection Network produces per-timestep, per-sensor attention weights at every inference call. The VSN-SHAP correlation of $\rho = 0.802$, verified in Part 3.1, is empirical evidence that the attention mechanism reflects true feature importance rather than learned artifacts. The VSN heatmap in the Technician view surfaces these weights directly.

Decision level. Can a specific recommendation be traced to input values and logic that a domain expert can evaluate? The Llama 3.2 advisory converts the top-5 perturbation sensitivity scores and current RUL context into a plain-language diagnostic. The digital thread preserves the full chain from sensor window through prediction, epistemic flag, scheduler output, and operator response, so any recommendation can be reconstructed from its inputs.

Governance level. Can this component be audited? Training data sources, architecture decisions, validation methodology, and performance metrics are documented in model metadata files with commit hashes and quality badges. The frozen backbone principle, the ESTIMATED badge policy, the dual normalization strategy, and the convergence stall detection logic are all preserved as documented design decisions rather than embedded conventions. An external auditor has access to the same evidence the system relies on internally.

The Twin as Federation: Governance by Scope

Part 1's scope section introduced the architecture's modularity. The governance framework formalizes this as a federation of nested twins, each with distinct ownership, state lifecycle, and validation scope.

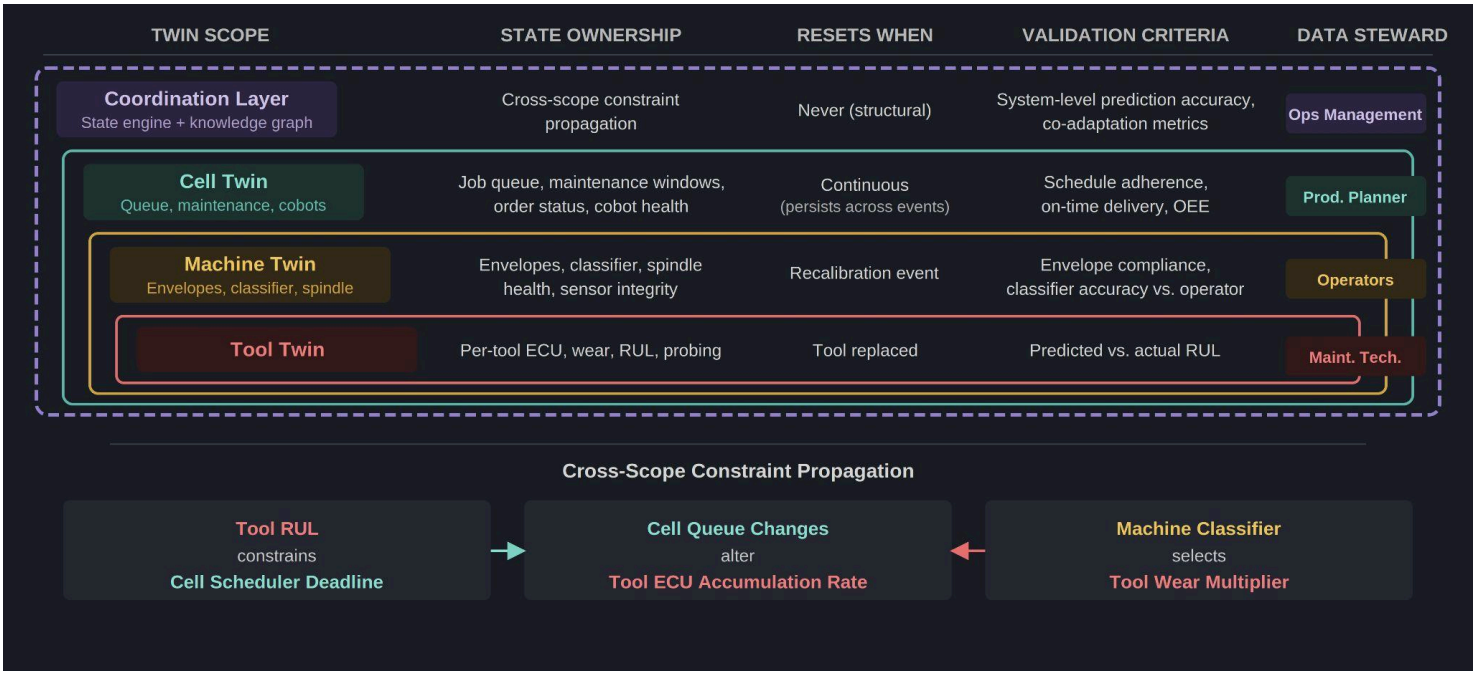


Figure 4.1. Federated twin governance hierarchy. Four nested scopes (tool, machine, cell, coordination layer) with columns for state ownership, reset conditions, validation criteria, and data steward. Arrows show cross-scope constraint propagation.

Twin Scope	State Ownership	Resets When	Validation Criteria	Data Steward
Tool twin	Per-tool ECU, wear segmentation, RUL, probing history	Tool replaced	Predicted vs. actual RUL at replacement	Maintenance technician
Machine twin	Envelopes, classifier, spindle health, sensor integrity	Recalibration event	Envelope compliance rate, classifier accuracy vs. operator verification	Operators
Cell twin	Job queue, maintenance windows, order status, cobot health	Continuous (persists across tool/machine events)	Schedule adherence, on-time delivery, OEE	Production planner
Coordination layer	Cross-scope constraint propagation via state engine and knowledge graph	Never (structural)	System-level prediction accuracy, co-adaptation metrics	Operations management

This framing clarifies a central governance question: who owns what, what resets when, and where coordination is required versus where twins operate independently. The scope boundaries reflect the physical structure of the system being twinned. A tool twin resets at replacement because the physical tool is gone. The coordination layer never resets because the relationships it encodes persist across any individual component's lifecycle.

Organizational Roles for Sustained Operation

The federated twin scopes assign data stewardship per scope. Three named roles operate across scopes to sustain the system as a whole, each with responsibilities that span multiple twins in the federation.

Model owner. Accountable for the twin's fidelity overall. Ensures the maintenance loop runs, manages calibration and upgrade cycles, maintains version history, and is the named point of accountability when outputs are questioned. The model owner holds the Validation and Monitoring links of the accountability chain. In practice this is a role within operations management rather than a separate position, but the accountability must be specifically assigned rather than left diffuse.

Data steward. Responsible for the data pipeline: sensor health, replacements, ingestion infrastructure, and continuity of the historical record across sensor generations. The data steward role is scope-specific in the federated architecture: the maintenance technician is the data steward for tool twins, operators are the data stewards for machine twins, and so on. The role's core responsibility is ensuring that sensor replacement and recalibration events are logged in the digital thread with enough context that downstream models can reason across the boundary rather than treating it as a distribution shift.

Operations liaison. Bridge between the twin and its users. Collects feedback on where outputs are trusted or ignored, communicates limitations to stakeholders who do not interact with the dashboard directly, and translates operational needs into model improvement requirements. The co-adaptation metrics in the Ops Manager view provide this role with quantitative evidence: declining override rates with sustained effectiveness indicate growing trust; declining overrides with declining effectiveness indicate the Level 3 slide described below. The operations liaison is the role most responsible for detecting uncritical acceptance before it becomes a governance failure.

Human-in-the-Loop Oversight

When system conditions deviate from defined bounds, the digital twin proposes decisions but does not act. The five feedback channels defined in Part 1 and implemented through the API endpoints in Part 3.1 each produce a structured record that the model consumes: ground-truth labels from maintenance confirmation, retraining data from anomaly acknowledgment, replanning triggers from status overrides, reward signals from job completion, and constraint updates from shift context. The prescriptive layer's autonomy boundary is defined by the epistemic state.

Epistemic State	OODA Autonomy Level	System Role	Human Role
NORMAL	Level 4 (Autonomous)	Observes, orients, decides, acts within defined bounds	Active tasks: scan entry, maintenance confirmation, shift logging. Not monitoring.
SENSOR_DRIFT_SUSPECTED	Level 2 (Recommend)	Observes and orients; flags sensor hardware for inspection	Decides whether to inspect, recalibrate, or override
DEGRADATION_CONFIRMED	Level 2 (Recommend)	Observes, orients, presents diagnostic context and recommendation	Decides maintenance timing and scope
UNCERTAIN_AMBIGUOUS	Level 2 (Recommend)	Observes, orients, presents competing hypotheses with costs	Decides which hypothesis to act on; resolves ambiguity

This dynamic autonomy is the architecture's answer to the Level 3 Trap. The OODA-based autonomy framework defines five levels, from fully manual through fully autonomous. Level 3, supervised autonomy, assigns all four OODA phases to the system while designating the human as a monitor. The trap is that monitoring a system operating correctly 99.9% of the time is not a meaningful task. The human's attention degrades precisely because the system rarely needs it, and when the 0.1% event arrives, the human is mentally absent. Level 3 is the design no-man's-land: if the system is reliable enough to run unsupervised, the argument for Level 4 is strong; if it is not, the argument for Level 2 is stronger.

The digital twin avoids Level 3 by never placing the operator in a passive monitoring role. During NORMAL operation, the system runs at Level 4; the operator is free for active tasks because the system does not need watching. During degraded states, the system drops to Level 2; the operator is engaged because the system has presented a decision that requires judgment, not a status indicator that requires vigilance. The five feedback channels reinforce this design. Each channel is an active task: the operator is teaching the system, and the system is learning from the operator. The co-adaptation metrics in the Ops Manager view track whether this engagement is sustained or eroding toward the behavioral signature of a Level 3 slide.

Personal Use and Non-Production Jobs

In many machine shops, machinists use cell equipment for personal projects outside production hours. This practice builds skill, maintains engagement with the equipment, and is part of the culture of craft that makes experienced machinists valuable. A digital twin that prohibits personal use, or that treats it as an anomaly to be flagged, misunderstands its relationship to the people it serves.

Personal use does create an epistemic gap. A tool used for a personal job accumulates wear the production wear model has not observed. If the system's RUL estimate assumes it has seen the tool's full history, and it has not, the estimate is confidently wrong by exactly the amount of untracked wear: the epistemic failure pattern from Part 1 applied to a social rather than a sensor-level cause.

The governance response is scaffolding, not prohibition. Personal jobs are permitted for approved users during designated non-production windows. The scaffolding requires a pre-job and post-job tool scan, either in-situ probing or optical measurement, both logged through the existing Technician view scan entry workflow. The delta between the two measurements is attributed as a non-production wear event, preserving the wear model's continuity without requiring the system to observe the personal job's cutting conditions. The system tracks the asset's condition rather than the person's activity.

If a personal job produces wear exceeding a configurable threshold relative to remaining tool life, the system flags the tool for review before production resumes. If the post-job scan is not completed, the system withholds RUL predictions for that tool until the scan is entered, displayed as a data-quality indicator rather than an error: *the system needs your input to give you a reliable prediction.*

Digital-Physical Synchronization

Personal use is a specific instance of a general problem: physical changes to the cell that occur without the twin's knowledge. A tool replaced during a weekend shift without correct logging. A fixture adjusted. An out-of-spec coolant added. Each alters the physical system that the twin models without updating the model's state.

Three detection mechanisms address this.

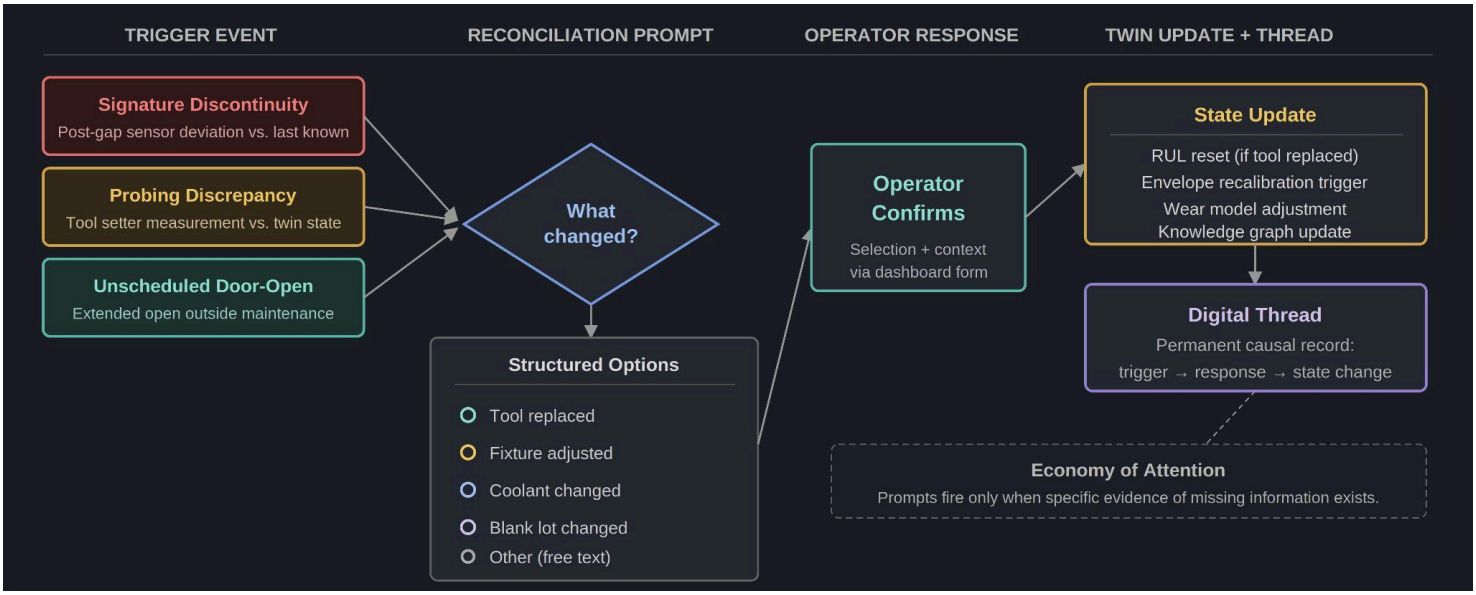


Figure 4.2. Digital-physical reconciliation flow. Trigger (signature discontinuity, probing discrepancy, or unscheduled door-open event) → reconciliation prompt with structured options → operator response → twin state update → digital thread record.

Signature discontinuity detection fires when the sensor signature on the first production cycle after a gap deviates significantly from the last recorded signature before the gap. In-situ probing at shift boundaries measures every tool in the magazine and compares results against the twin's last known state; discrepancies surface immediately. The door interlock described in Part 2.2 logs extended door-open events outside scheduled maintenance windows, linked to subsequent signature changes in the causal chain.

The design principle governing these mechanisms is economy of attention. Every prompt competes for the operator's cognitive bandwidth. A system that demands constant manual entry trains its users to dismiss prompts reflexively, which is worse than no prompts at all. Reconciliation prompts fire only when the system has specific evidence that information is missing, and the prompt shows what was detected and why clarification is needed. Over time, operators learn to log changes proactively. The scaffolding teaches the behavior it eventually makes unnecessary.

This is the upskilling premise underlying the entire governance framework. The digital twin does not replace the machinist's knowledge of the equipment. It gives that knowledge a place to live beyond the individual: in the digital thread, in the knowledge graph, in the wear models that learn from every logged event. An experienced machinist who adjusts a fixture based on thirty years of intuition is performing an act of expertise. The twin captures the outcome of that expertise so that the next machinist inherits a system that has learned from every predecessor's judgment. How long before the accumulated institutional memory in the twin begins to shape training for new machinists, and what does that feedback loop look like when it closes?

Part 4.3: Equity, Privacy, and Cybersecurity

Equity and Fairness

The system must not produce systematically worse predictions for one product type, one shift, one tooling configuration, or one operator's work pattern than for another.

Equity Risk	Description	Mitigation
Synthetic-to-real gap	Robot joint model trained on simulation may underperform on real RTDE data; R^2 = 0.9955 is an upper-bound estimate	ESTIMATED badge policy; explicit upper-bound caveat in every view
Domain overconfidence	Operators may treat ESTIMATED domain outputs as ground truth	Badge system + epistemic flag distinguish validated from estimated accuracy
Alert threshold inequity	Default RUL threshold (30 cycles) may be too conservative for some products, too aggressive for others	Hot-reloadable per deployment; operator feedback enables asset-specific adaptation
Operator skill gap	Different technical backgrounds may lead to misinterpretation of uncertainty indicators	Four stakeholder views calibrate depth to audience; Llama advisory provides plain-language translation

A subtler concern involves the co-adaptation dynamics inherent in the feedback architecture. If one operator's conservative replacement behavior dominates calibration-period training data, the model learns that operator's risk tolerance rather than the tool's actual wear trajectory. Subsequent operators inherit a biased model. The survival analysis methods in Part 3.1 mitigate this by modeling operator behavior as a variable rather than adopting it as ground truth; the co-adaptation metrics in the Ops Manager view monitor whether this bias is developing.

Privacy and Data Minimization

The system handles no personally identifiable information. Sensor readings are machine-level signals. The technician ID field is a role identifier used for audit trail purposes, provided voluntarily at maintenance confirmation. Data minimization is structural: only the fourteen features required for inference are stored in the pipeline, and raw sensor readings follow the three-tier retention schedule described in Part 4.2. The local-first architecture provides a privacy guarantee independent of policy compliance: all inference, advisory generation, and data storage run on local hardware, and no data leaves the facility network.

Cybersecurity

The entire inference pipeline runs on local hardware with no cloud dependency. The system can operate on an air-gapped network, eliminating the threat class associated with cloud communication: data exfiltration, man-in-the-middle interception, remote API exploitation, and dependency on external service availability. This is a structural security guarantee. After initial setup, the system requires no internet connectivity.

An air-gapped system eliminates external attack vectors. It does not eliminate internal ones. In a facility with restricted network access, the most realistic threat is an insider: sabotage by a disgruntled employee, corporate espionage targeting process parameters, or unauthorized modification of system behavior by someone with physical or credential-based access.

Detection Layer	Threat Vector Addressed	Mechanism
Digital thread (forensic)	Post-incident reconstruction of tampering events	Permanent, non-deletable causal records with operator identity, timestamps, job context. Full chain from anomalous reading through prediction, alert, and operator response.
Epistemic integrity (statistical)	Deliberate sensor manipulation	Out-of-distribution detection regardless of cause. Cross-sensor consistency requires simultaneous coordinated manipulation of multiple independent physical signals to evade detection.
Knowledge graph (structural)	Falsified records, unauthorized tool substitution, product misidentification	Causal inconsistencies above the sensor level: tool wearing inconsistently with product history, classification contradicting active NC program.
Configuration version match (access control)	Unauthorized model weight substitution	Version mismatch warning before inference proceeds. Production deployment places configuration endpoints behind role-based access control mapped to federated twin governance.

The architecture's honest limitation: sabotage that operates below the sensor layer's detection threshold. Gradual manipulation designed to stay within cross-sensor correlation expectations, or physical interference with the machining process itself (contaminated coolant, loosened fixtures, substituted blank material) that degrades quality without producing out-of-distribution sensor signatures, would not trigger the epistemic integrity layer because the signals would appear physically consistent. Detection of these vectors requires the quality inspection processes outside the twin's system boundary, reinforcing the scope decision from Part 1. In the context of insider threat, expansion to include downstream quality inspection moves from desirable to important.

Part 4.4: Communication and Impact

Communication Strategy

The same sensor data, flowing through the same models, must produce different outputs for different audiences because each audience makes decisions at a different timescale. The four stakeholder views described in Part 3.2 implement this through temporal lens calibration rather than information hiding. Three confidence indicators persist across all views: model confidence, health score, and the four-state epistemic flag, all as primary visual elements. When the epistemic state changes, every view reconfigures around the system's current level of self-knowledge, as documented in the state-transition figures in Part 3.2.

Visualization Refinements

Three visualization decisions were refined across the project's development based on what was learned about stakeholder decision needs.

The tool life display evolved from static percentage bars to interactive drill-down panels. The confidence intervals in the expanded detail are the answer to a question the static bars could not address: not just how much life remains, but how certain the estimate is.

The Planner view was redesigned around the before/after Gantt comparison because this is the question production managers ask when an alert fires: not "what is the RUL?" but "what does maintenance cost me in schedule disruption, and what does skipping it risk?" The demonstration scenario illustrates the structure with a

representative failure event; the actual savings magnitude will be determined by the cell's specific failure frequency and schedule density.

The VSN attention heatmap surfaces the model's internal reasoning as a 30-cycle × 14-sensor color grid. Its reliability depends on the empirically verified VSN-SHAP correlation of ρ = 0.802, which ensures the heatmap reflects true feature importance rather than learned artifacts.

Impact Metrics and Success Evaluation

Impact Level	Timescale	Metric	Measurement
Prediction accuracy	Cycle to shift	RUL RMSE / nRMSE per domain	Validated on benchmark datasets (CMAPSS, NUAA, TUAWS, NIST); operational accuracy established as cell data accumulates
Prediction accuracy	Cycle to shift	Product classification accuracy	Validated at 99.93% on TUAWS; operational accuracy tracked via operator verification at changeover
Alert quality	Shift to week	False alarm rate	Operator-marked false alarms ÷ total alerts per rolling window; target < 20%
Scheduling impact	Shift to week	Makespan delta per maintenance event	Before/after Gantt comparison quantifies cost of intervention vs. cost of failure
Operational performance	Week to quarter	OEE, unplanned downtime, tooling cost/part, on-time delivery	Tracked in Ops Manager view; demonstration values illustrate form; actuals emerge from operational history
System trust	Quarter to year	Override rate, override effectiveness, scan compliance, system acceptance	Co-adaptation metrics tracking joint operating point between operators and system
Model convergence	Quarter to year	Wear multiplier CV, prediction error trend	Convergence stall detection triggers structural review

The algorithm benchmarks establish the system's capability on representative data. They do not constitute performance guarantees on this specific cell. The twinnable architecture is system-agnostic: the same inference pipeline, model structure, and feedback mechanisms apply to any compatible sensor suite. The transition from benchmark accuracy to operational accuracy begins when the system ingests real sensor data and accumulates the replacement events, operator labels, and calibration scans that replace training-set priors with asset-specific parameters. This convergence is the mechanism by which assumed confidence gives way to measured confidence. The rate is set by the slowest-accumulating feedback channel: the per-product replacement events that feed the wear multiplier recalibration. And the process has a floor. Beyond a certain density of operational evidence, further replacement events stop improving the multiplier estimates, and further accuracy gains require new data sources, such as in-process metrology or downstream quality inspection, rather than more operational history.

The digital twin's value proposition, stated in Part 1 and developed through every subsequent section, is the translation between observation and decision. The metrics above measure whether that translation is accurate, timely, and trusted. But the deepest measure of success is one the metrics cannot directly capture: whether the people who use the system understand what it knows, what it does not know, and where the boundary between those two conditions sits. The four-state epistemic flag, the replacement-event validation loop, the co-adaptation metrics, and the convergence floor are the mechanisms that keep that boundary visible. They do not guarantee the system will never be wrong. They guarantee that when it is wrong, the architecture makes the error detectable, and when it is uncertain, the architecture says so. Whether that guarantee holds across the operational lifetime of the twin, not just at deployment, is the question the governance framework, the validation plan, and the feedback architecture exist to answer.

References

ACADEMIC REFERENCES

Lim, B., Arık, S. Ö., Loeff, N., & Pfister, T. (2021). Temporal fusion transformers for interpretable multi-horizon time series forecasting. International Journal of Forecasting, 37(4), 1748–1764. https://doi.org/10.1016/j.ijforecast.2021.03.012

Nie, Y., Nguyen, N. H., Sinthong, P., & Kalagnanam, J. (2023). A time series is worth 64 words: Long-term forecasting with transformers. In Proceedings of the 11th International Conference on Learning Representations (ICLR 2023). https://arxiv.org/abs/2211.14730

Saxena, A., Goebel, K., Simon, D., & Eklund, N. (2008). Damage propagation modeling for aircraft engine run-to-failure simulation. In 2008 International Conference on Prognostics and Health Management (pp. 1–9). IEEE. https://doi.org/10.1109/PHM.2008.4711414

Sayyad, S., Kumar, S., Bongale, A., Kotecha, K., & Abraham, A. (2023). Remaining useful-life prediction of the milling cutting tool using time–frequency-based features and deep learning models. Sensors, 23(12), 5659. https://doi.org/10.3390/s23125659

Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., & Long, M. (2023). TimesNet: Temporal 2D-variation modeling for general time series analysis. In Proceedings of the 11th International Conference on Learning Representations (ICLR 2023). https://arxiv.org/abs/2210.02186

Zerveas, G., Jayaraman, S., Patel, D., Bhamidipaty, A., & Eickhoff, C. (2021). A transformer-based framework for multivariate time series representation learning. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD '21) (pp. 2114–2124). https://doi.org/10.1145/3447548.3467401

DATASET CITATIONS

Agogino, A., & Goebel, K. (2007). Milling data set [Data set]. BEST Lab, UC Berkeley. NASA Prognostics Data Repository, NASA Ames Research Center. https://www.nasa.gov/intelligent-systems-division/discovery-and-systems-health/pcoe/pcoe-data-set-repository/

Engelmann, B., Schmitt, A.-M., & Martinez, M. (2025). Production data set for five-axis CNC milling with multiple changeovers (Version 1.0.1) [Data set]. Zenodo. https://doi.org/10.5281/zenodo.15735480

Li, Y., Liu, C., Li, D., Hua, J., & Wan, P. (2021). Tool wear dataset of NUAA_Ideahouse [Data set]. IEEE Dataport. https://doi.org/10.21227/3aa1-5e83

Saxena, A., & Goebel, K. (2008). Turbofan engine degradation simulation data set [Data set]. NASA Prognostics Data Repository, NASA Ames Research Center. https://www.nasa.gov/intelligent-systems-division/discovery-and-systems-health/pcoe/pcoe-data-set-repository/

Scania CV AB. (2017). APS failure at Scania trucks [Data set]. UCI Machine Learning Repository. https://doi.org/10.24432/C51S51

Weiss, B. A. (2018). Process and robot data from a two robot workcell representative of manufacturing operations [Data set]. National Institute of Standards and Technology. https://doi.org/10.18434/mds2-2361

Process Note

Claude (Anthropic) was used throughout research, writing, prototyping, and implementation. Architecture, design decisions, and system specification are the project team's; the AI served as a collaborative tool in developing and expressing them. Dashboard mockups were prototyped with Claude's assistance from layouts and specifications defined by the project lead. The cover image was generated using Google Gemini. The twinnable software system was implemented through a multi-agent stack combining Claude and Codex, guided by the lead software engineer.