

A Dual-Tier RFID Architecture for Mold Lifecycle and Production Control in Polyurethane Manufacturing: A Simulation-Based Evaluation

Quang Vinh Tran

Ho Chi Minh City University of Technology: VNUHCM-Ho Chi Minh City University of Technology

THI DIEM CHAU LE

lechau@hcmut.edu.vn

Ho Chi Minh City University of Technology: VNUHCM-Ho Chi Minh City University of Technology
<https://orcid.org/0000-0003-4045-0981>

Research Article

Keywords: RFID, Mold lifecycle management, Discrete-event simulation, Production interlock, Polyurethane manufacturing, Industry 4.0

Posted Date: October 8th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-10794787/v1>

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

A Dual-Tier RFID Architecture for Mold Lifecycle and Production Control in Polyurethane Manufacturing: A Simulation-Based Evaluation

Quang Vinh Tran^{1,2} and Thi Diem Chau Le^{1,2,*}

¹ School of Industrial Management, Ho Chi Minh City University of Technology (HCMUT), 268 Ly Thuong Kiet Street, Dien Hong Ward, Ho Chi Minh City, 70000, Vietnam

² Vietnam National University Ho Chi Minh City, Vo Truong Toan Street, Linh Xuan Ward, Ho Chi Minh City, Vietnam

* Corresponding author: Thi Diem Chau Le; Email: lechau@hcmut.edu.vn

Abstract In polyurethane (PU) casting lines, precise tracking of mold lifecycles governs both product quality and planning efficiency. Manual monitoring produces systematic count discrepancies and offers no mechanism to prevent overproduction or the continued use of structurally expired molds. This study proposes a dual-tier radio frequency identification (RFID) architecture in which a work-order execution tier bounds output by the planned quota while a lifecycle tier tracks each mold against a structural limit of 600 passes. Validating a pass requires the elapsed curing interval to fall inside a calibrated time gate and the accumulated count to remain below the quota. Because full-scale deployment is capital-intensive and disruptive to a continuous line, the architecture is evaluated ex ante through a stochastic discrete-event simulation of a ten-line, 120-mold casting shop, benchmarked against the incumbent manual policy over 25 replications. The simulation reveals a result not evident from the specification alone: time-gated validation on strict two-reader pairing attains only 90.26% (± 0.07) mold-level pass-count accuracy at realistic on-metal detection probabilities, below the manual baseline of 93.52% (± 0.04), because single-sided read losses are discarded as exceptions. Adding a finite-state recovery layer that reconstructs single-sided detections raises accuracy to 98.06% (± 0.03) and holds it above 96% even at a per-reader detection probability of 0.90. Over a 120-day horizon the recovered architecture reduces expired-mold exposure by 59.3%, defects from expired molds by 75.7%, overproduction by 78.9% and planning labour by 90.4%. Sensitivity analysis further requires oven residence-time standard deviation to be held below approximately 1.0 min.

Keywords RFID, Mold lifecycle management, Discrete-event simulation, Production interlock, Polyurethane manufacturing, Industry 4.0

1 Introduction

In the era of Industry 4.0, the integration of cyber-physical systems and the Industrial Internet of Things plays a central role in raising production efficiency [1], [2]. For manufacturing plants that operate through tooling, and particularly for polyurethane (PU) casting lines, the precise management of mold lifecycles and cycle times determines in both product quality and operational productivity [3], [4].

Many facilities still rely on manual data collection to track the number of production passes executed by each mold. This approach carries three significant limitations.

First, manual counting is prone to systematic discrepancy. Because tally sheets are reconciled at the line rather than the individual mold, cumulative life is typically imputed by dividing batch yield among the molds in circulation. Since molds are dedicated to different item families and therefore carry persistently different daily work orders, this allocation introduces a per-mold bias rather than zero-mean noise. Molds that have exceeded their standard lifespan, 600 passes in the plant considered here, consequently remain in production, while other molds are withdrawn with usable life remaining. Continued use of degraded molds directly produces visual defects on finished bushes, including cracked bushes, missing pin holes, inclined shells, and abnormal surface conditions arising from excessive or insufficient release agent.

Second, the absence of real-time data significantly prolongs production planning. At the reference plant, engineering and management staff consume in excess of 90 labour-hours per week processing backlogged mold and yield data before a daily plan can be issued.

Third, adherence to the planned output quantity rests on manual vigilance alone, so casting can continue past the work order and generate overproduction waste that lean manufacturing practice specifically seeks to eliminate [5]. The bound proposed in this study therefore operates at the level of the individual mold: once the validated pass count of a mold reaches its planned quota, the system rejects further validations for that mold and signals the operator to withdraw it from circulation, while the casting line and the curing oven continue to run. Halting the casting equipment itself is neither required nor economical in a continuous PU process, where a stop–restart cycle costs more than the overproduction it prevents.

To address these limitations, this study proposes a management architecture based on RFID. While RFID is widely adopted for logistics and inventory tracking [6], [7], its application in harsh manufacturing environments involving high temperatures and metallic interference requires specific architectural adaptation [8], [9]. In the proposed system, high-temperature anti-metal tags are affixed to each mold, and reader–antenna assemblies are deployed at the poly pouring (casting) and de-mold stations. The architecture separates two concerns that conventional

enterprise systems conflate: execution of the current work order, and management of the structural life of the tool.

A practical constraint shapes the methodology of this study. Physical deployment across a continuous casting shop requires antenna mounting on oven doors, tagging of every mold, and middleware integration with the plant database; committing that capital before the control logic has been shown to meet its accuracy target exposes the project to substantial rework risk. This risk is not hypothetical: metallic detuning is known to degrade read performance severely for tags mounted on metal objects [8], [9], and passive UHF tags exposed to sustained industrial heat require explicit thermal characterisation and modified packaging before they can be relied upon outside their nominal range [10]. The architecture is therefore evaluated *ex ante* by stochastic discrete-event simulation, an approach that is well established both for evaluating RFID-enabled process redesign before adoption [11] and as the first stage of simulation-led digital twin development [13]–[15]. Accordingly, this manuscript reports simulated rather than field-measured performance; the field validation protocol required to close that gap is specified in Section 6.

The main contributions of this study are:

- An RFID architecture adapted to PU casting, combining a work-order execution tier with a mold lifecycle tier, in which pass validation is gated jointly on the thermal curing window and on the residual planned quota.
- An *ex ante* evaluation framework based on stochastic discrete-event simulation, permitting the architecture, its control logic, and its sensitivity to physical-layer degradation to be assessed before capital commitment.
- A quantified counter-intuitive finding: strict two-reader temporal pairing, the configuration most directly implied by the literature on temporal filtering of RFID event streams, is inferior to manual counting at realistic on-metal detection probabilities. We identify the responsible mechanism, which is the asymmetric discarding of single-sided reads, and propose a finite-state recovery layer that resolves it.
- A set of deployment requirements derived from sensitivity analysis, expressed as bounds on per-reader detection probability, curing-gate width, and oven residence-time variability, which the physical installation must satisfy in order to meet the $\geq 95\%$ accuracy target.

The remainder of this study is organised as follows. Section 2 reviews related work on RFID in industrial asset management and identifies the research gap. Section 3 presents the system architecture and control design. Section 4 describes the simulation model and experimental design. Section 5 analyses the results. Section 6 concludes and sets out directions for future work.

2 Related Work

The transition toward Industry 4.0 has accelerated adoption of RFID for traceability in smart manufacturing. Applying these paradigms to semi-continuous chemical processes such

as PU casting, however, requires established models to be adapted to specific operational and environmental constraints.

2.1 RFID-Enabled Real-Time Manufacturing Execution

A substantial body of work has established RFID as the data-acquisition layer of real-time manufacturing execution systems. Zhong et al. [1] deployed RFID devices systematically across a mass-customisation shop floor to track manufacturing objects and collect real-time production data, demonstrating that planning and scheduling decisions can be made and executed more precisely once shop-floor uncertainty is observable. The same research programme extended this into a two-level advanced planning and scheduling model for RFID-enabled ubiquitous manufacturing [16], while Dai et al. [3] reported an equivalent architecture in an automotive part manufacturer. Fang et al. [4] proposed an event-driven multi-agent platform organising real-time RFID events into abstraction levels suitable for work-in-progress decisions, and Barenji et al. [17] developed a multi-agent RFID-enabled distributed control system for a flexible manufacturing shop. More recently, de Jesus Pacheco et al. [2] documented a longitudinal in-depth case study of an RFID-based monitoring and optimisation system built on open-source and low-cost technologies, exposing the empirical difficulties of implementing intelligent manufacturing systems in practice.

These contributions establish the feasibility of RFID-driven execution control. The behaviour of the control logic when the physical detection layer degrades, however, remains unaddressed in this body of work, and it is the central concern of the present study.

2.2 Temporal Filtering of RFID Event Streams

A primary challenge in industrial RFID deployment is the generation of false-positive reads and signal bounces in environments dominated by metal machinery and high-temperature ovens. Feng et al. [18] applied a big-data approach to RFID-captured shop-floor data, deriving process times and failure rates per operator and process type, and identified bottlenecks and quality inefficiencies directly from the captured event stream. Ertek et al. [19] formalised the mining of RFID data from schedule-based systems, where the expected timing of events supplies the constraint against which observed events are validated.

The common principle is temporal: an RFID event pair, for example moving-in and moving-out of a workstation, is considered valid only if the interval between the sequential detection nodes t_i and t_j satisfies a predefined constraint,

$$T_{min} \leq t_j - t_i \leq T_{max} \quad (1)$$

While effective for discrete assembly logistics, this bound requires precise calibration for chemical manufacturing. The present methodology inherits this foundation and adapts it into a rigid time-gated validation mechanism calibrated to

the standardised thermal curing window of PU materials. Critically, and as Section 5 demonstrates, the literature is largely silent on what happens when one of the two detection nodes fails to register: the validity test in (1) is undefined for a single-sided observation, and the default behaviour of discarding such events proves to be the dominant error source in this application.

2.3 Cycle-Based Tool Life and Predictive Maintenance

Conventional tooling management estimates tool lifespan from aggregated batch yields, an approach that does not account for the actual physical stress cycles borne by an individual tool. Analysis of RFID-captured production data has shifted the focus toward cycle-based remaining useful life (RUL) estimation.

The underlying damage model is not new. The linear damage accumulation hypothesis, introduced by Palmgren for rolling bearings [20] and generalised by Miner [21], holds that each load cycle consumes a fixed fraction of total life and that failure occurs once the accumulated fractions reach unity. Writing the accumulated damage of the Palmgren–Miner rule as a residual against a structural cycle limit gives the RUL of an industrial asset subject to repetitive stress as

$$RUL(k) = L_{max} - \sum (i = 1 \dots k) \omega_i \cdot c_i \quad (2)$$

where L_{max} is the ultimate structural limit, c_i represents individual operational cycles, and ω_i is a wear-weighting factor corresponding to the damage fraction of the Palmgren–Miner formulation. The present study operationalises this model through RFID tracking, standardising each successfully validated casting pass as a uniform degradation unit ($\omega_i = 1$) in order to compute the remaining lifespan of individual PU molds. Setting $\omega_i = 1$ is the constant-amplitude special case of [21], and is defensible here because every casting pass subjects the mold to the same thermal and mechanical cycle; Section 6 identifies its relaxation as future work. Section 6 identifies relaxation of the uniform-wear assumption as a direction for future work.

2.4 Cyber-Physical Production Control and Lean Interlocks

The integration of RFID within cyber-physical systems enables real-time interaction between physical shop-floor assets and digital control architectures [17], [22]. To prevent manufacturing deviation, such control must enforce a dynamic constraint on cumulative output,

$$\forall t : Q_{actual}(t) \leq Q_{target}(t) \quad (3)$$

Lean manufacturing practice emphasises the systematic elimination of overproduction waste, and traditional lean instruments suffer from high latency in high-speed environments. Brintrup et al. [5] developed an opportunity-analysis toolset that maps RFID deployment onto the seven Toyota Production System wastes, identifying automated data capture, assurance of data dependencies, and improved

production and inventory visibility as the mechanisms through which RFID contributes to a leaner manufacturing environment. Despite this foundation, a gap remains in the use of RFID networks to procedurally lock operations on the shop floor. The present study bridges that gap by transforming the theoretical capacity constraint (3) into an active automated interlock that restricts physical casting once the planned target P_{PLAN} has been achieved.

2.5 Environmental Constraints on RFID in PU Casting

Deploying RFID in PU casting lines presents severe physical-layer challenges. Molds are constructed from dense metal, which reflects and absorbs radio-frequency energy and causes read-range degradation, data collision, and blind spots. Thirappa et al. [8] addressed this class of problem through a slim tag antenna designed specifically for metallic tools with a narrow footprint, and Yeap et al. [9] quantified the adverse effect of a surrounding metal environment on passive tag readability. In addition, PU processing requires molds to circulate continuously inside a curing oven. In the facility studied, the oven operates at approximately 90 °C, and the mold remains within it on both the loaded and the return leg of its circuit, so a tag mounted on the mold is held near that temperature for essentially the whole production shift; accumulated over the 600-pass structural limit this amounts to on the order of 475 hours of exposure. That operating point lies above the +85 °C nominal continuous rating typical of general-purpose passive UHF label inlays, so an off-the-shelf adhesive tag is not admissible and the deployment requires an on-metal tag qualified for sustained operation above that range. Seré et al. [10] show that operating a commercial passive UHF tag beyond its nominal temperature range requires explicit thermal characterisation and modified packaging, which indicates that thermal endurance rather than peak survival is the governing tag-selection criterion and that it must be resolved before deployment rather than assumed. These findings motivate the parameter ranges over which detection probability is varied in Section 5.3.

2.6 Research Gap

Three gaps follow from the foregoing. First, an integrated architecture that combines precise mold pass counting, real-time cycle-time monitoring, and continuous environmental data collection, and that also correlates specific mold usage thresholds with the prevention of specific visual quality defects, remains absent from the literature. Second, the temporal-filtering literature specifies when an event pair is valid but not how the counting policy should behave under single-sided detection loss, which is the normal condition on metallic tooling. Third, RFID execution architectures are typically reported after deployment, so their sensitivity to physical-layer degradation is rarely quantified in advance, even though that property determines whether a given hardware specification will meet its accuracy target. This paper addresses all three.

3 System Architecture and Control Design

The proposed system is organised around three modules: physical hardware deployment, dual-tier control logic, and real-time data synchronisation.

3.1 Hardware Deployment

Identification devices. High-temperature, anti-metal RFID tags are permanently affixed to each mold. Each tag carries a uniquely encoded identifier (Mold Code) linked to a centralised database.

Data collection nodes. The production process operates as a closed loop. To calculate cycle times accurately,

antenna and reader assemblies are deployed at two bottleneck nodes:

- Inbound station (casting area - Reader In), located where the polyurethane mixture is poured into the mold. This node initiates the cycle counting sequence.
- Outbound station (de-mold area - Reader Out), located at the oven exit where finished products are extracted. Data captured at this node closes the time cycle and validates a production pass.

Fig. 1 presents the physical arrangement, comprising the extended curing oven with entry- and exit-mounted readers, the dual-tier tracking logic, and the production interlock acting on the de-mold decision.

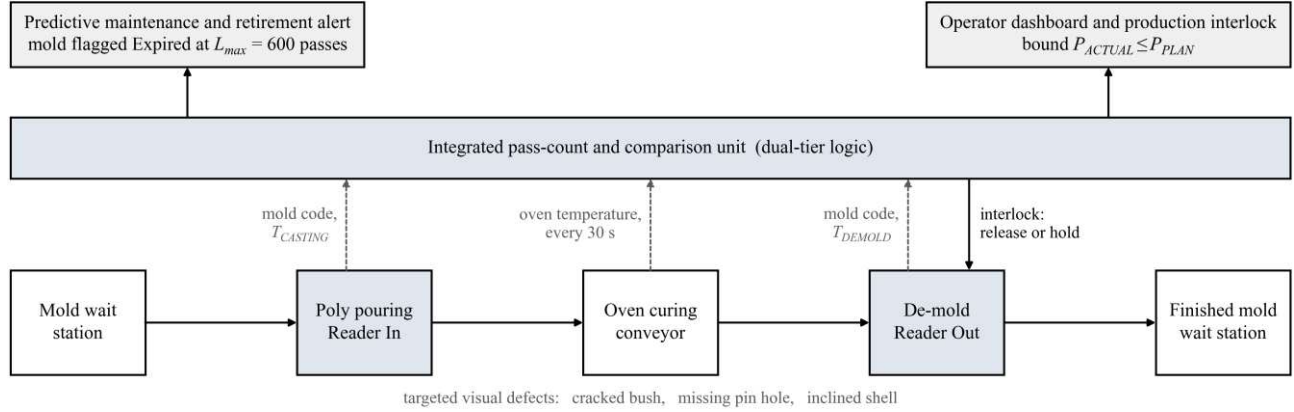


Fig. 1 Architecture of the instrumented polyurethane casting line. Material flows along the station row; RFID readers at poly pouring and de-mold, together with the oven temperature sensors, feed the integrated pass-count and comparison unit, which drives the retirement alert and the production interlock.

3.2 Dual-Tier Tracking Algorithm

Unlike conventional enterprise resource planning systems, the proposed architecture separates daily production tracking from overall mold lifecycle management. The logic relies on three parameters: the planned passes for the current work order (P_{PLAN}), the actual validated passes completed in real time (P_{ACTUAL}), and the maximum structural lifespan of the mold (L_{max}), standardised at 600 passes.

Tier 1 - work-order execution. To simplify operator decision-making, the real-time status of a mold is evaluated against the daily target using a binary classification:

- **IN PROCESS** when $P_{ACTUAL} < P_{PLAN}$. The mold has not met its quota and continues in the casting loop.
- **DONE** when $P_{ACTUAL} = P_{PLAN}$. The target is fulfilled, and the system activates an overproduction interlock. The visual status switches to DONE, signalling the operator to extract the mold. The system thereby bounds the production logic to $P_{ACTUAL} \leq P_{PLAN}$; subsequent scans that would push the mold beyond P_{PLAN} are rejected, preventing unauthorised surplus casting.

Tier 2 - mold lifecycle. Each validated pass decrements the remaining useful life computed by (2). When the residual life reaches zero, the mold is flagged Expired and must be removed from the line for refurbishment. Because Tier 2

acts on the same validated event stream as Tier 1, any systematic bias in validation propagates directly into the retirement decision. Section 5 quantifies the consequences of that coupling.

3.3 Time-Gated Validation

Let $T_{CASTING}$ be the timestamp recorded at the inbound reader and T_{DEMOLD} the timestamp at the outbound reader. The elapsed curing time is $\Delta T = T_{DEMOLD} - T_{CASTING}$. A physical pass is validated and added to P_{ACTUAL} if and only if two conditions hold simultaneously: the elapsed time falls within the standardised PU curing window, and the accumulated count remains strictly below the planned quota:

$$(25 \text{ min} \leq \Delta T \leq 30 \text{ min}) \wedge (P_{ACTUAL} < P_{PLAN}) \quad (4)$$

If a mold is detected at the outbound station with $\Delta T < 25$ min, or if P_{ACTUAL} has already reached P_{PLAN} , the event is flagged as an anomaly and ignored. This mechanism eliminates duplicate scans, premature mold extraction, and overproduction in a single test.

3.4 Single-Sided Read Recovery

Condition (4) is defined only for an event pair. On metallic tooling passing through a curing oven, however, detection at either node may fail, and the resulting single-sided observation is undefined under (4) and therefore discarded.

Because the test can only reject, its errors are one-directional and accumulate as a systematic undercount.

The architecture is therefore augmented with a finite-state recovery layer. Each mold carries a state flag named IN-OVEN. A pass is validated if any of the following holds:

- Both readers detect the mold and ΔT lies inside the gate, which is the nominal case of (4).
- Only Reader Out detects the mold, IN-OVEN is set, and the interval since the last recorded de-mold lies inside the gate, recovering a lost inbound read.
- Only Reader In detects the mold, and the mold is subsequently detected inbound again after an interval consistent with one complete cycle, giving a deferred inference of a lost outbound read.

Rule 3 succeeds with probability p_{rec} , reflecting cases in which deferred inference is ambiguous because the mold

was withdrawn from the line between passes. Section 5 quantifies the contribution of this layer, which proves to be decisive.

3.5 Operator Dashboard and Environmental Synchronisation

The real-time dashboard displays, for each active mold, the Mold ID, P_{PLAN} , P_{ACTUAL} , and the binary execution status, allowing operators to execute mold transitions at the moment a unit reaches DONE. Fig. 2 summarises the reference-period output of the plant's existing monitoring system, giving planned and good quantities together with oven temperature for each active line. These figures supply the baseline production and quality parameters used in Section 4 and the face-validity check reported in Section 4.7.

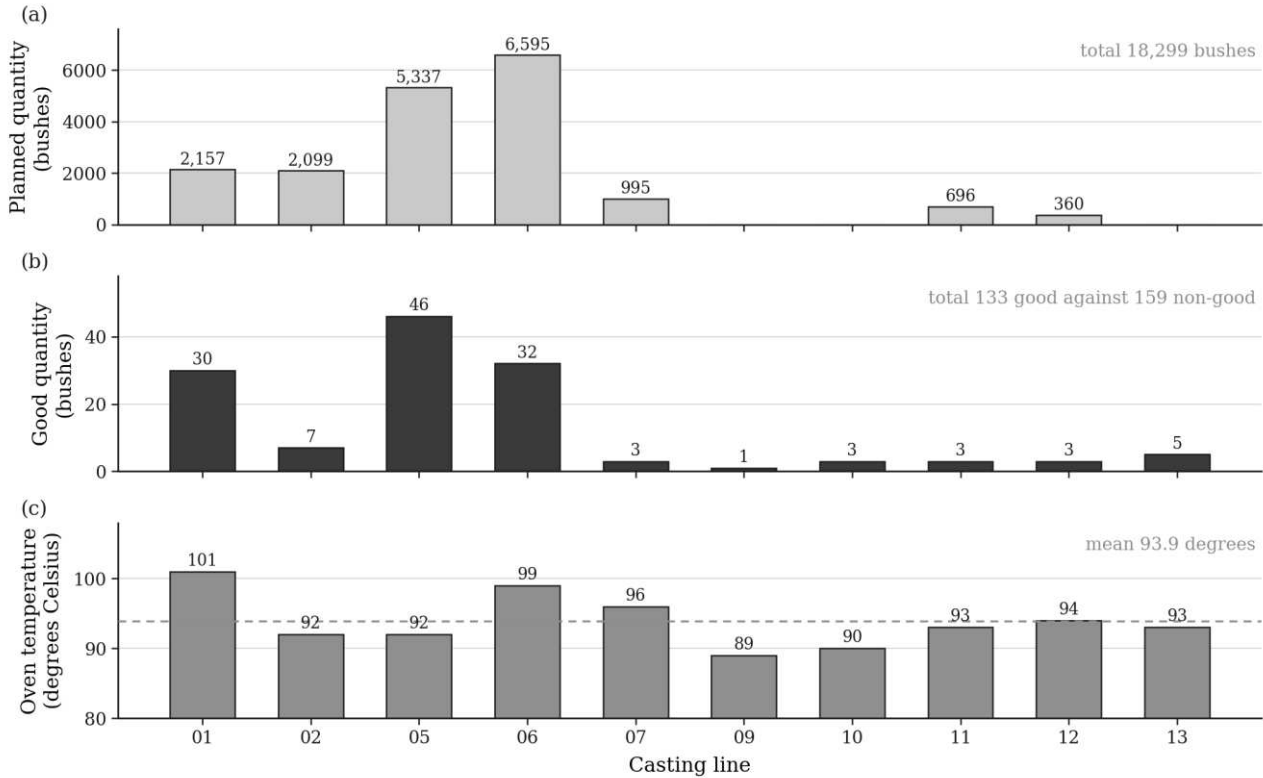


Fig. 2 Reference-period production data of the plant, by casting line: (a) planned quantity, (b) good quantity, (c) oven temperature, with the mean shown dashed. Values are those recorded on the plant monitoring dashboard and supply the baseline parameters of Section 4.4.

Temperature sensors integrated along the oven transmit data to the central dashboard at least once every 30 s. Synchronising this continuous temperature record with the pass timestamp of individual molds enables engineers to trace root causes when finished products exhibit surface defects, release-agent anomalies, or dimensional deviation. This channel forms part of the architecture but is not exercised by the present model, which treats curing temperature as implicit in the residence-time distribution; Section 5.8 notes the consequence.

4 Simulation Model and Experimental Design

4.1 Rationale for an Ex Ante Model-Based Evaluation

Discrete-event simulation (DES) is the established instrument for evaluating manufacturing system designs before commitment, and its use to assess RFID-enabled process redesign prior to adoption isprecedented [11]. Where such a redesign has instead been evaluated after deployment, RFID-derived location data has been used to measure productivity before and after the change [12]; that

approach establishes the operational relevance of the redesign question but does not provide an ex ante instrument for it. Recent work positions DES as the analytical core of simulation-led digital twin development, in which an offline model is built first and incrementally connected to live data as the physical system comes online [14], [15]. The model presented here is intended to serve that role: an offline decision-support model now, and the kernel of an online digital twin after deployment.

4.2 Conceptual Model and System Boundary

The model represents one PU casting shop containing $N_L = 10$ active lines, each with $N_M = 12$ molds in circulation, giving 120 tracked molds. Each mold executes a closed-loop cycle: mold wait, poly pouring at Reader In, oven curing conveyor, de-mold at the oven exit under Reader Out, then back to mold wait. A single traversal is one physical pass.

The oven circuit is bidirectional and the mold is inside the oven on both legs, but only the loaded leg between Reader In and Reader Out is subject to the validation gate; the unread return leg is therefore represented by the aggregate mold-wait term rather than modelled separately, since it carries no detection event.

The model maintains two distinct counters per mold:

- $P_{true}(m)$, the ground-truth cumulative pass count, visible only to the simulator.
- $P_{sys}(m)$, the count held by the tracking system, on which the shop-floor policy actually acts.

The divergence between these counters is the primary object of study, since it is what drives premature or delayed mold retirement and hence the visual defects the system is intended to prevent. Fig. 3 shows the model structure.

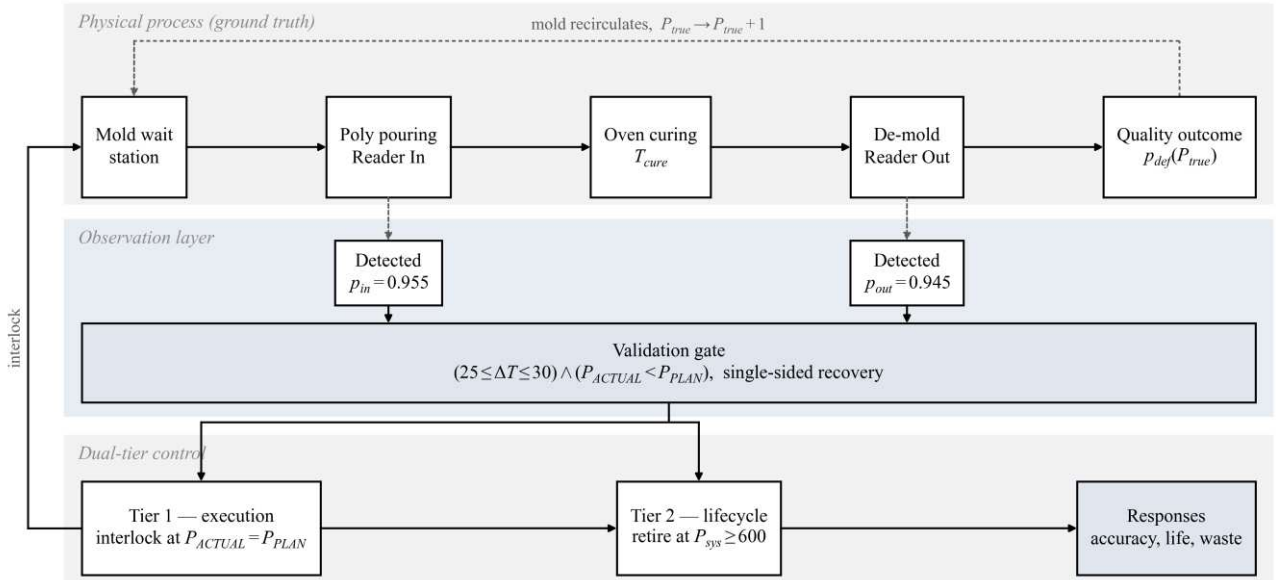


Fig. 3 Conceptual structure of the simulation model, showing the physical process layer that generates ground truth, the observation layer that degrades it through imperfect detection and gate rejection, and the dual-tier control layer that acts on the resulting system count.

The system boundary excludes upstream material preparation, downstream packing, and mold refurbishment logistics. Machine breakdown and operator absence are excluded, so that the model isolates the effect of the tracking and control policy while holding the physical process constant across policies. This restriction is deliberate and is revisited in Section 5.8.

4.3 Control Policies Compared

Four policies are simulated. P1 and P3 correspond respectively to the incumbent state and to the architecture as specified in Sections 3.2 and 3.3; P2 and P4 are ablation and enhancement variants that isolate the contribution of each mechanism.

Table 1 Control policies under comparison

ID	Policy	Pass validation rule	Interlock	Mold life estimation
P1	Manual	Operator tally per shift	Soft (operator discretion)	Batch yield allocated equally over line molds
P2	RFID naive	Any Reader-Out detection	None	Per-mold RFID count
P3	RFID gated	$read_{in} \wedge read_{out} \wedge (25 \leq \Delta T \leq 30)$	Hard, $P_{ACTUAL} < P_{PLAN}$	Per-mold RFID count
P4	RFID gated + recovery	P3, plus single-sided read recovery (Section 3.4)	Hard	Per-mold RFID count

Policy P1 reproduces the mechanism identified in Section 1 as the root cause of imprecise lifecycle management: because manual records attribute output at the line level, cumulative life is imputed to individual molds by dividing batch yield by the number of molds in circulation. Since molds carry persistently different daily work orders, this

allocation carries a systematic per-mold bias. High-utilisation molds accumulate unrecorded wear and overrun $L_{\max}$; low-utilisation molds are retired prematurely with usable life remaining.

4.4 Stochastic Inputs

Table 2 Model input parameters

Parameter	Symbol	Value	Distribution	Provenance
Oven residence time	T_{cure}	27.5 ± 0.85 min	Normal, truncated [24, 31]	Conveyor set-point; calibrate from timestamp logs
Pouring time	-	2.4 ± 0.5 min	Normal, truncated [1, 6]	Time study
De-mold time	-	3.1 ± 0.7 min	Normal, truncated [1, 8]	Time study
Inter-cycle queueing	-	4.6 ± 1.8 min	Normal, truncated [0.5, 15]	Time study
Reader-In detection probability	p_{in}	0.955	Bernoulli	Calibrate from tag trial
Reader-Out detection probability	p_{out}	0.945	Bernoulli	Calibrate from tag trial
Spurious read probability per cycle	p_{fp}	0.021	Bernoulli	Site survey
Manual omission probability	-	0.062	Bernoulli	Tally-versus-output reconciliation
Manual double-entry probability	-	0.024	Bernoulli	As above
Manual shift-total bias	-	$\pm 4.5\%$	Normal	As above
Baseline visual-defect probability	p_0	0.0082	Bernoulli	Plant NG records
Wear-induced defect slope	k	4.2×10^{-4} per pass beyond $L_{\max}$	-	Regression of NG rate on mold age
Structural mold limit	$L_{\max}$	600 passes	Deterministic	Engineering standard
Daily work order per mold	P_{PLAN}	8–17 passes, persistent per mold	Uniform with daily jitter	Production plan
Shift length	-	480 min	Deterministic	-

Visual-defect probability for a mold that has executed P_{true} passes is modelled as

$$p_{\text{def}}(P_{\text{true}}) = \min(0.35, p_0 + k \cdot \max(0, P_{\text{true}} - L_{\max})) \quad (5)$$

This functional form is not constructed ad hoc. It is the linear deterioration case of the imperfect-production-process literature, in which a manufacturing system operates in an in-control state until a shift occurs and thereafter produces defective items at a rate that grows with elapsed production. Rosenblatt and Lee [23] introduced this structure and derived optimal production cycles under constant, linear, and exponential deterioration; Kim and Hong [24] generalised it to arbitrary shift-time distributions and confirmed the linear case as the standard intermediate model. Equation (5) is that linear case, with the state shift located deterministically at the structural cycle limit $L_{\max}$ rather than at a random time, which is appropriate because mold degradation past its rated life is a physical certainty rather than a stochastic event. The mechanism it represents is the one by which degradation past the rated cycle limit produces cracked bushes, missing pin holes, and inclined shells at an accelerating rate.

The slope k and the base rate p_0 are the two parameters of this form, and both must be estimated from plant NG records before the absolute defect figures of Section 5.2 carry weight; the relative ranking of the four policies is insensitive

to them, since all four are evaluated under the same quality model.

4.5 Performance Measures

Let D be the set of mold-days simulated, $P_{\text{true}}(m, d)$ the physical passes executed by mold m on day d , and $P_{\text{rec}}(m, d)$ the count registered by the tracking system. Mold-level pass-count accuracy is

$$A_{\text{mold}} = 100 \cdot [1 - (1/|D|) \cdot \sum_{\text{over } (m, d) \in D} |P_{\text{rec}}(m, d) - P_{\text{true}}(m, d)| / P_{\text{true}}(m, d)] \quad (6)$$

Equation (6) is the mean absolute percentage error (MAPE) of the recorded count against the true count, expressed as an accuracy rather than an error. MAPE is scale-independent, which is required here because daily work orders vary between molds, and it is directly interpretable by production staff as a percentage. Hyndman and Koehler [25] survey this class of measure and recommend the mean absolute scaled error over MAPE on three grounds: MAPE is undefined at zero denominators, is heavily skewed when the actual value is near zero, and is asymmetric in that it penalises positive errors more heavily than negative errors of equal magnitude. The first two objections do not bind here, since mold-days with no physical pass are excluded from D . The asymmetry objection does apply and is not resolved by the present

choice; MAPE is retained for its direct interpretability on the shop floor, and the comparison between policies remains valid because every policy is scored with the same measure against the same true counts.

A_{mold} is reported in preference to the line-aggregate equivalent A_{lines} , because mold-level accuracy governs remaining useful life estimation. Line-aggregate accuracy is optimistically biased: independent per-mold omissions and double counts partially cancel on aggregation, which is precisely the illusion that makes manual line-level reporting appear adequate while individual mold life estimates drift.

Additional measures are: retirement error, $|P_{\text{true}} - P_{\text{ret}}| / L_{\text{max}}$, decomposed into wasted life (retired early) and overrun life (retired late); expired-mold exposure, the number of passes physically executed with $P_{\text{true}} > L_{\text{max}}$; overproduction, bushes cast in excess of the work order; and planning labour, hours per week for data entry, shift consolidation, and exception handling.

4.6 Provenance of the Model Relations

Because the credibility of a simulation study rests on the provenance of its constituent relations as much as on its implementation, Table 3 states the origin of every equation used. Three categories are distinguished: relations adopted from established literature, relations that operationalise a literature model for the specific conditions of PU casting, and relations derived within this study from the model's own parameters.

Table 3 Origin of the model relations

Eq.	Quantity	Origin	Status
(1)	Temporal validity window for an RFID event pair	Spatial-temporal filtering of shop-floor RFID streams [15], [16]	Adopted
(2)	Cycle-based remaining useful life	Palmgren–Miner linear damage accumulation rule [17], [18]	Adopted; $\omega_i = 1$ is the constant-amplitude case
(3)	Cumulative output constraint	Lean e-Kanban overproduction bound [5]	Adopted
(4)	Joint gate and quota validation condition	Operationalises (1) and (3) for the PU curing window	This study
(5)	Defect probability as a function of mold wear	Linear deterioration case of the imperfect-production-process model [20], [21]	Adopted; shift point fixed at L_{max}
(6)	Mold-level pass-count accuracy	Mean absolute percentage error [22]	Adopted
(7)–(8)	Per-pass recording probability α	Elementary probability applied to the	Derived here

Eq.	Quantity	Origin	Status
		model's own parameters	
(9)	Accuracy under one-directional error	Follows from (6) and the interlock stopping rule	Derived here
(10)	Steady-state overrun at retirement	Follows from (2) and the definition of α	Derived here
(11)–(12)	Planning labour	Accounting identity over the model's activity counts	Derived here

Relations (7) to (12) are stated and verified in Section 5.6. They are not borrowed results, and they make no claim to novelty. They follow directly from the model's own definitions, and their function here is to make the simulated output auditable rather than to contribute theory.

4.7 Verification and Validation

The model was verified and validated following the structured procedure of Sargent [26].

Verification. Degeneracy and extreme-condition tests were executed. Setting $p_{\text{in}} = p_{\text{out}} = 1$ and $p_{\text{fp}} = 0$ drives $A_{\text{mold}} \rightarrow 100\%$ and expired exposure to its structural floor; disabling the interlock reproduces unbounded overproduction; widening the gate to $(-\infty, \infty)$ recovers the naive counting policy. Event-trace inspection confirmed correct ordering of pour, cure, and de-mold events and correct interlock firing at $P_{\text{ACTUAL}} = P_{\text{PLAN}}$.

Analytical cross-check. Beyond the degeneracy tests, the principal responses were predicted in closed form from the model's input parameters and compared against the simulated output. Agreement is within 0.3 percentage points for pass-count accuracy under the interlocked policies and within 0.1 h/week for the planning-labour model. The derivations and the full comparison appear in Section 5.6, since they also serve to explain the results rather than merely to verify them.

Face validity. At baseline parameters the model produces a visual-defect rate of 0.898% (± 0.010) under the manual policy. Scaled to the 18,299-bush monthly work order shown in Fig. 2, this corresponds to 164.2 NG bushes per month against the 159 NG bushes recorded on the plant dashboard for the reference period, a deviation of 3.3%. The model likewise reproduces the order of magnitude of planning effort reported by the engineering team, in excess of 90 h per week.

Statistical validity. Each configuration was run for 25 independent replications, and 15 for the extended-horizon and sensitivity experiments, with distinct random seeds over a 20-day production horizon. Molds were initialised with wear drawn uniformly from $[0, L_{\text{max}})$ to avoid a synchronised retirement transient, which removes the need for a separate warm-up period. Results are reported as means

with 95% confidence half-widths; all half-widths for accuracy measures are below 0.15 percentage points.

Limitation on validation. No operational validity test against field data has yet been performed, since the physical system is not deployed. The model is therefore validated for structure and behaviour, not for predictive accuracy. This is restated as a limitation in Section 5.8.

4.8 Simulation Environment and Reproducibility

The model is implemented in Python 3 with NumPy; SciPy is used only for the normal-quantile computations reported in Table 8. No commercial simulation package is required, which keeps the model portable and directly extensible by the plant engineering team.

Rather than a general-purpose event calendar, the implementation exploits the regular structure of the casting loop. Because each mold executes an independent sequence of pour–cure–de-mold cycles within a shift, all process variates for a production day are pre-generated as arrays, and the shift capacity of each mold is obtained from the cumulative sum of its cycle times. This vectorised formulation reduces one replication of the 120-mold, 20-day model to approximately 0.1 s on a standard workstation, so a 25-replication experiment completes in under 3 s and the full sensitivity study of Section 5 in under 2 min. That runtime is what makes the parameter sweeps of Tables VI–VIII practical at the replication counts required for the reported confidence intervals.

Variance reduction is obtained through common random numbers. All physical-process variates, covering pouring, curing, de-mold and queueing times as well as defect realisations and reader detection outcomes, are drawn from a single primary stream consumed in identical order under every control policy, so the four policies are evaluated on the same realisation of the physical process and the comparison is paired. Policy-specific nuisance variates, namely the shift

batch-allocation noise that applies only to the manual policy, are drawn from an independent secondary stream so that they cannot desynchronise the primary stream. Every replication is seeded deterministically, and all results reported in this paper are therefore exactly reproducible. The model source, the experiment driver that regenerates every table and figure, and the resulting data files are provided as supplementary material.

4.9 Use of Large Language Models

The authors derived the research problem, two-tier architecture, model logic, and parameter values shown in Table 2 from real-world enterprise data and developed the experimental design. Additionally, Claude Opus 5 (Anthropic) served as a technical assistant in three respects. First, it implemented, to the authors' specification, the discrete-event simulation model described in this section, the analysis scripts that produced the results reported in Section 5, and the plotting code that renders Figs. 1, 3, 4, 5 and 6 from the reported data; those figures are deterministic outputs of that code, no figure in this article is a generative-AI image, and Figs. 2 and 7 were drawn by the authors. Second, it assisted in drafting and language-editing the text of this manuscript, which the authors then revised. Third, it was used to check the bibliographic metadata of the reference list against Crossref and Semantic Scholar records and the alignment between in-text claims and the content of the cited sources. The authors verified the implementation through the degeneracy tests and the independent analytical cross-check reported in Section 5.6, reviewed and edited all generated content, and take full responsibility for the model, its results and the content of this research.

5 Results and Discussion

5.1 Baseline Comparison of Control Policies

Table 4 Performance of the four control policies (10 lines, 120 molds, 20 production days, 25 replications; mean $\pm$ 95% confidence half-width)

Policy	A_{mold} (%)	A_{line} (%)	Retirement error (%)	Expired-mold passes	Overproduced bushes	Planning labour (h/week)
P1 Manual	93.52 $\pm$ 0.04	96.06 $\pm$ 0.05	2.41 $\pm$ 0.09	573 $\pm$ 35	868 $\pm$ 35	90.1 $\pm$ 0.2
P2 RFID naive	94.32 $\pm$ 0.05	96.47 $\pm$ 0.06	1.53 $\pm$ 0.06	407 $\pm$ 23	833 $\pm$ 34	4.2 $\pm$ 0.0
P3 RFID gated	90.26 $\pm$ 0.07	89.99 $\pm$ 0.07	2.80 $\pm$ 0.08	734 $\pm$ 30	865 $\pm$ 35	27.0 $\pm$ 0.2
P4 RFID gated + recovery	98.06 $\pm$ 0.03	97.99 $\pm$ 0.03	1.21 $\pm$ 0.05	313 $\pm$ 16	183 $\pm$ 8	8.6 $\pm$ 0.1

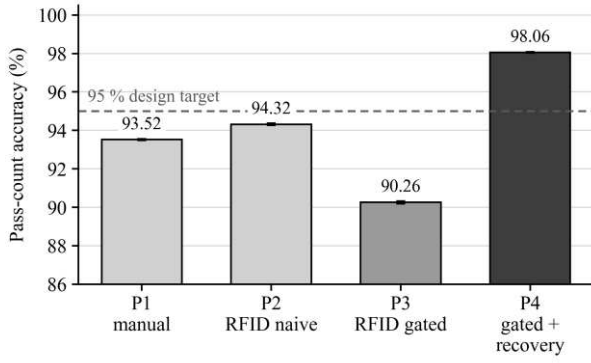


Fig. 4 Mold-level pass-count accuracy of the four control policies, with 95% confidence intervals. The dashed line marks the 95% design target.

Three observations follow.

First, the gated architecture as specified in Section 3.3 underperforms the manual baseline. Policy P3 attains 90.26% mold-level accuracy against 93.52% for P1. The mechanism is asymmetric, and it can be accounted for in closed form. The conjunction $\text{read}_{\text{in}} \wedge \text{read}_{\text{out}}$ succeeds with probability $p_{\text{in}} \cdot p_{\text{out}} = 0.9025$, and the curing time falls inside the gate with probability 0.9968, so a pass is recorded with probability $\alpha = 0.8996$: approximately one pass in ten is discarded as an exception even though it was physically completed. Because the shift terminates on the recorded count, the mold executes P_{PLAN} / α passes for every P_{PLAN} recorded, which places accuracy at $100 \alpha = 89.96\%$, within 0.3 percentage points of the simulated 90.26%. Section 5.6 sets out the derivation. Because the gate can only reject, its errors are one-directional and accumulate as a systematic undercount, whereas manual omissions and double entries partially offset one another. This is the principal finding of the study. It cannot be deduced from the architectural specification, and emerges only once the physical-layer detection probability is modelled explicitly.

Second, the undercount propagates into worse lifecycle management, not better. Because P_{sys} lags P_{true} by roughly 10% under P3, molds are retired late, and expired-mold exposure rises to 734 passes against 573 under manual

tracking. A tracking system that systematically undercounts is more dangerous than no tracking system at all, since operators trust its output.

Third, the recovery layer resolves the problem completely. Policy P4 raises accuracy to 98.06%, reduces retirement error to 1.21%, and cuts overproduction by 78.9%, because the interlock now fires at the correct count rather than late, relative to manual operation. The gain comes from two additional acceptance paths: an outbound-only detection recovers 0.0424 of passes and an inbound-only detection a further 0.0378, lifting α from 0.8996 to 0.9798 and hence accuracy from 89.96% to 97.98% by relation (9). Planning labour falls from 90.1 to 8.6 h per week, a 90.4% reduction. The manual figure is the product of roughly 1360 mold-day records at 27 s each plus 0.78 h of shift consolidation per line, giving 18.0 h per production day. The RFID figure is dominated by the exception term, which scales with $(1 - \alpha)$: P4 raises about 28 exceptions per day against 137 for P3, which is why the more restrictive design costs three times the labour.

Policy P2 achieves 94.32% accuracy at only 4.2 h/week of labour, appearing superficially competitive. Its apparent advantage over P1 is an artefact of error cancellation rather than of better measurement: its per-pass recording probability of 0.9660 is barely higher than the manual 0.9620, and both benefit from two-directional errors that partially offset within the day. It is in any case unacceptable, since it admits approximately 580 spurious reads per horizon as valid passes, provides no interlock, and leaves overproduction at 833 bushes, statistically indistinguishable from manual operation. Automation without validation transfers the error rather than removing it.

Fig. 7 makes the underlying mechanism visible. It traces the true and system-held cumulative pass counts of a single high-utilisation mold over 90 production days under three policies. Under P1 and P3 the system count drifts progressively below the true count, so the mold reaches approximately 670 physical passes before the system registers the 600-pass limit. Under P4 the two traces remain nearly coincident and the mold is retired at approximately 615 passes.

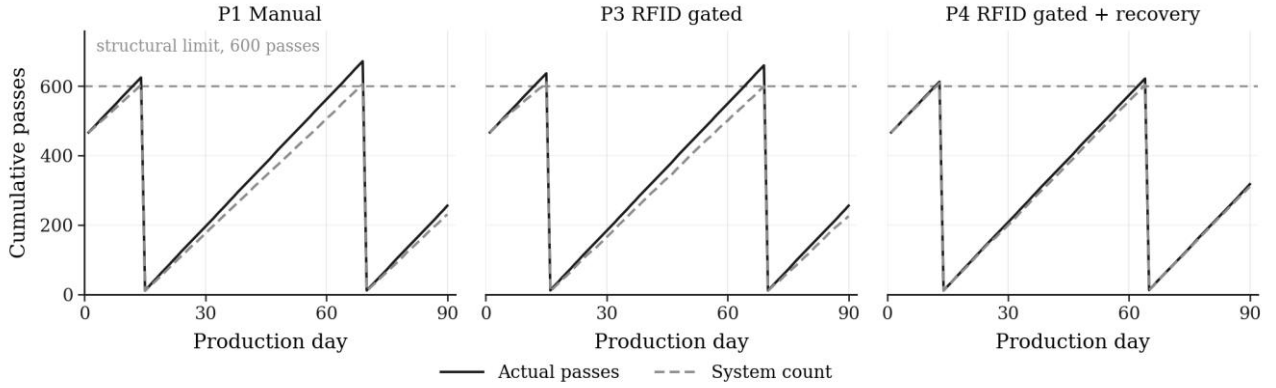


Fig. 7 True versus system-held cumulative pass count for one high-utilisation mold (work order 17 passes per day) over 90 production days. The dotted line marks the 600-pass structural limit. The sawtooth discontinuities are mold retirements and replacements.

5.2 Mold Lifecycle and Quality Impact over an Extended Horizon

Over a 20-day horizon the difference in overall visual-defect rate between policies is not statistically resolvable, because

expired-mold passes constitute only about 2% of total output. The horizon was therefore extended to 120 production days, approximately six months, with 15 replications.

Table 5 Extended-horizon results (120 production days, 15 replications)

Policy	A_{mold} (%)	Expired-mold passes	Defects on expired molds	Overall defect rate (%)	Overproduced bushes
P1 Manual	93.57 ± 0.04	9475 ± 251	201.8 ± 6.3	0.898 ± 0.010	5235 ± 285
P2 RFID naive	94.30 ± 0.03	5630 ± 71	78.9 ± 4.1	0.840 ± 0.010	5001 ± 274
P3 RFID gated	90.25 ± 0.05	13974 ± 159	303.3 ± 10.3	0.939 ± 0.012	5168 ± 268
P4 RFID gated + recovery	98.04 ± 0.02	3858 ± 63	49.1 ± 4.6	0.831 ± 0.010	1105 ± 52

Relative to the manual baseline, policy P4 reduces expired-mold exposure by 59.3%, defects attributable to expired molds by 75.7%, overall visual-defect rate by 7.5%, and overproduction by 78.9%. The confidence intervals for defects on expired molds are disjoint between P1 and P4, so the quality effect is statistically resolvable at this horizon. The overall defect-rate improvement, while consistent in sign across every replication, is smaller in relative terms because most defects arise from causes unrelated to mold age and are unaffected by the tracking policy. Claiming a large headline reduction in total NG rate would therefore be unsupported, and no such claim is made.

The decomposition of retirement error is informative. Under manual tracking, 2.07% of mold life is discarded through premature retirement while a further 2.54% is consumed past L_{max} through delayed retirement. Both failure modes occur at once and for the same reason, namely that batch-yield allocation cannot distinguish high-utilisation from low-utilisation molds. The magnitude follows directly from the allocation arithmetic. A mold running 17 passes per day is credited with the line mean of 12.5, an effective α of 0.74, so it reaches 816 physical passes before the records show 600, an overrun of 216. A mold running 8 passes per day is credited with 12.5, an effective α of 1.56, and is scrapped at 384 physical passes with 216 passes of usable

life discarded. Averaged over the work-order distribution, this produces the simultaneous waste and overrun observed. RFID-based per-mold counting eliminates premature retirement entirely across all three RFID policies, since P_{sys} can only undercount; the residual problem is confined to overrun, and its magnitude tracks the accuracy of the counting rule, at 2.81% for P3 against 1.23% for P4.

5.3 Sensitivity to Physical-Layer Detection Probability

The dominant uncertainty in deployment is the detection probability actually achievable with anti-metal tags on hot steel molds. Detection probability was varied from 0.90 to 0.99 per reader.

Table 6 Pass-count accuracy versus reader detection probability (15 replications per point)

$P_{\text{in}} / P_{\text{out}}$	P3 gated, A_{mold} (%)	P4 gated + recovery, A_{mold} (%)
0.900 / 0.890	80.24 ± 0.13	96.02 ± 0.07
0.920 / 0.910	83.86 ± 0.13	96.81 ± 0.06
0.940 / 0.930	87.49 ± 0.12	97.54 ± 0.05
0.955 / 0.945	90.28 ± 0.09	98.05 ± 0.04
0.970 / 0.960	93.07 ± 0.08	98.54 ± 0.04

p_{in} / p_{out}	P3 gated, A_{mold} (%)	P4 gated + recovery, A_{mold} (%)
0.990 / 0.980	96.86 $\pm$ 0.06	99.14 $\pm$ 0.03

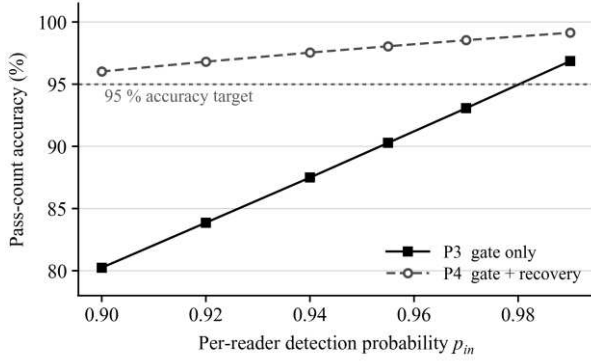


Fig. 5 Mold-level pass-count accuracy as a function of reader detection probability, for gate-only validation and for gated validation with single-sided read recovery. The dotted line marks the 95% accuracy target.

The two policies exhibit qualitatively different robustness. P3 degrades approximately linearly because, by relations (7) and (9), its accuracy is simply $100 \cdot p_{in} \cdot p_{out} \cdot P(\text{gate})$; at equal per-reader probability p this is close to $100p^2$, so a one-point loss per reader costs roughly two points of accuracy. It fails to reach the 95% target anywhere below $p \approx 0.98$, a specification that is not realistically attainable on metallic molds passing through a curing oven [8], [9]. P4 remains above 96% across the entire range tested, including the pessimistic case of 0.90, because its recovery paths contribute $2p(1 - p)$ of additional acceptance probability, a term that grows precisely as the conjunction term shrinks.

This yields a direct engineering implication. Under P3, meeting the accuracy target would require capital expenditure on redundant antennas, higher-grade tags, and RF shielding to push detection probability toward 0.98. Under P4 the same target is met by a middleware change at essentially zero hardware cost. The accuracy requirement is therefore satisfied more cheaply in software than in hardware. This conclusion could not have been reached without the model, and it materially changes the procurement specification.

5.4 Sensitivity to Curing-Gate Width

Table 7 Effect of validation gate width (policy p4)

Gate [T_{min} , T_{max}] (min)	A_{mold} (%)	Rejected reads per horizon	Planning labour (h/week)
[26.0, 29.0]	91.08 $\pm$ 0.06	2502	25.0
[25.5, 29.5]	96.61 $\pm$ 0.06	941	12.0
[25.0, 30.0]	98.05 $\pm$ 0.04	539	8.6
[24.0, 31.0]	98.36 $\pm$ 0.04	453	7.9
[23.0, 32.0]	98.36 $\pm$ 0.04	453	7.9

Narrowing the gate below the nominal [25, 30] window is sharply counterproductive: a [26, 29] window rejects valid passes whose curing time lies in the tails of the residence-time distribution, dropping accuracy by seven percentage points and tripling exception-handling labour. The magnitude is again predictable. With residence time distributed $N(27.5, 0.85)$, a [26, 29] window captures 92.9% of passes against 99.7% for [25, 30]; multiplying the recovery-adjusted acceptance probability by these two figures gives 91.0% and 97.7% accuracy, against 91.08% and 98.05% simulated. Widening beyond [24, 31] produces no further gain. This saturation is partly an artefact of the truncation bounds imposed on the residence-time distribution, and in any case the practical upper bound on gate width should be set by the process requirement that a genuinely under-cured pass must be rejected, not by count accuracy. The nominal [25, 30] window is confirmed as a reasonable operating point, but the analysis shows it should be treated as a calibrated parameter derived from the measured residence-time distribution rather than as a fixed material constant.

5.5 Sensitivity to Oven Residence-Time Variability

Because the gate acts on the measured curing interval, its rejection rate is governed by the dispersion of oven residence time.

Table 8 Effect of oven residence-time variability (policy p4, gate [25, 30])

$\sigma(T_{cure})$ (min)	Passes inside gate (%)	A_{mold} (%)
0.60	100.00	98.36 $\pm$ 0.04
0.85	99.67	98.05 $\pm$ 0.04
1.20	96.28	94.89 $\pm$ 0.07
1.60	88.18	87.31 $\pm$ 0.07
2.20	74.42	74.42 $\pm$ 0.09

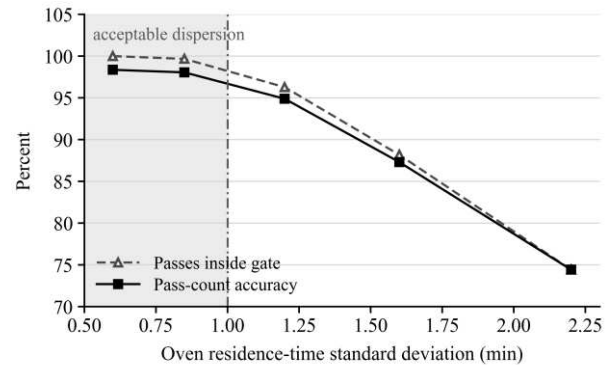


Fig. 6 Mold-level pass-count accuracy and share of passes falling inside the validation gate, as a function of oven residence-time standard deviation. The dash-dotted line marks the resulting process specification limit.

Accuracy tracks the in-gate share almost exactly, which is what relation (8) predicts: with the recovery layer in place

the acceptance probability is dominated by the gate factor $P(\text{gate})$, so $A_{\text{mold}} \approx 0.9829 \cdot P(\text{gate}) \cdot 100$. At $\sigma = 2.20$ min the gate admits only 74.4% of passes and accuracy collapses to the same figure. Gate rejection, rather than read loss, has become the binding constraint. The practical specification follows directly: the conveyor oven must hold residence-time standard deviation below approximately 1.0 min for the tracking system to meet a 95% accuracy target under a [25, 30] min gate.

This converts an information-system requirement into a process-control requirement on conveyor speed regulation and oven door dwell. Without the model, such a requirement would surface only after installation, as an unexplained accuracy shortfall. It also indicates a design alternative worth evaluating: widening the gate in proportion to measured dispersion, at the cost of weakened protection against under-cured passes.

5.6 Analytical Verification of the Simulated Results

The headline figures of Sections 5.1 to 5.5 are not opaque outputs of a black box. Each follows, to first order, from a small number of closed-form relations, and the agreement between those relations and the simulated values constitutes a strong verification test in the sense of Sargent [26]. This subsection derives the principal quantities and compares them against the simulation.

1) Per-pass recording probability. Define α as the probability that one physical pass is recorded by the tracking system. Since the reader detections and the curing-time realisation are independent,

$$\alpha P3 = p_{\text{in}} \cdot p_{\text{out}} \cdot P(25 \leq T_{\text{cure}} \leq 30) = 0.955 \times 0.945 \times 0.9968 = 0.8996 \quad (7)$$

$$\alpha P4 = \alpha P3 + (1 - p_{\text{in}}) \cdot p_{\text{out}} \cdot P(\text{gate}) + p_{\text{in}} \cdot (1 - p_{\text{out}}) \cdot p_{\text{rec}} = 0.8996 + 0.0424 + 0.0378 = 0.9798 \quad (8)$$

For the manual policy the expected increment per pass is $1 - 0.062 + 0.024 = 0.9620$, and for the naive RFID policy it is $p_{\text{out}} + p_{\text{fp}} = 0.9660$. The three RFID values differ only in how single-sided detections are treated, which isolates the effect of the recovery layer at 0.0802 in recorded probability.

2) Mold-level accuracy. The two families of policy require different treatment, and the distinction explains the central result of Section 5.1.

Under the interlocked policies P3 and P4 the counting error is strictly one-directional: the gate can reject a valid pass but never invent one, and the shift terminates when the recorded count reaches the quota. The mold therefore executes P_{PLAN} / α physical passes while the system records P_{PLAN} , so

$$A_{\text{mold}} = 100 \cdot (1 - |P_{\text{PLAN}} - P_{\text{PLAN}}/\alpha| / (P_{\text{PLAN}}/\alpha)) = 100 \quad (9)$$

Under P1 and P2 the errors run in both directions, through omissions and double counts under manual tally and through missed reads and spurious reads under naive RFID, so a proportion of them cancel within the day. A comparable closed form is unavailable for these two policies; a normal approximation to the stopping-time error supplies a lower bound, and the simulation resolves the exact value. This asymmetry is the reason a policy with a smaller net bias (P3 at -10.04%) can nevertheless be more accurate than one with a larger apparent error rate, and conversely why P1, whose per-pass error rate of 8.6% exceeds P4's 2.02%, still outperforms the gate-only design.

3) Steady-state mold overrun. After each retirement both counters reset, so divergence accumulates from zero over a full lifecycle. The system retires the mold when its recorded count reaches L_{max} , by which time the mold has physically executed L_{max} / α passes:

$$\text{overrun} = L_{\text{max}} (1/\alpha - 1) \quad (10)$$

giving 67.0 passes for P3 and 12.4 for P4. Because retirement is evaluated at the end of each production day, the recorded count overshoots L_{max} by up to one day's output, which adds a further few passes.

4) Planning labour. Write n for the mean number of passes per mold-day. With N_L lines and N_M molds per line, manual effort is the record-entry burden plus per-line shift consolidation, whereas the RFID policies incur only exception handling plus dashboard review:

$$H_{\text{manual}} = N_L N_M n \cdot h_{\text{rec}} + N_L \cdot h_{\text{cons}} \quad (11)$$

$$H_{\text{RFID}} = N_L N_M n (1 - \alpha) \cdot h_{\text{exc}} + N_L \cdot h_{\text{rev}} \quad (12)$$

The exception term scales with $(1 - \alpha)$, which is why P3, despite being the more restrictive design, generates three times the exception workload of P4.

5) Overproduction under the interlock. With a hard interlock, output cannot exceed the quota as recorded; every overproduced bush is therefore an unrecorded pass. Two conditions must hold jointly, and the second is easily overlooked: the mold's work order must also fit inside the shift. Molds whose planned quantity exceeds what an 8-hour shift can physically deliver are limited by the shift, not by the interlock, and contribute nothing to overproduction. With a mean of 11.4 passes per mold-day against work orders spanning 8 to 17 passes, only about 40% of mold-days are quota-limited.

Table 9 Closed-form prediction versus simulated value

Quantity	Relation	Predicted	Simulated	Deviation
$A_{\text{mold}}, P3$	(9)	89.96%	90.26%	-0.30 pp
$A_{\text{mold}}, P4$	(9)	97.98%	98.06%	-0.08 pp
Overrun at retirement, P3	(10)	67.0 passes	67.1 passes	-0.1
Overrun at retirement, P4	(10)	12.4 passes	16.7 passes	-4.3
Planning labour, P1	(11)	90.1 h/week	90.1 h/week	0.0

Quantity	Relation	Predicted	Simulated	Deviation
Planning labour, P3	(12)	26.9 h/week	27.0 h/week	-0.1
Planning labour, P4	(12)	8.7 h/week	8.6 h/week	+0.1
Overproduction, P4, 120 days	Section 5.6-5	1351 bushes	1105 bushes	+246

Simulated overrun figures are taken from a 400-day horizon, at which the lifecycle reaches steady state; the 120-day values of Table 5 remain partly transient because molds are initialised with shared counters.

The agreement for the interlocked accuracies and for the labour model is within 0.3 percentage points and 0.1 h/week respectively, confirming that the implementation reproduces its intended logic. The two larger deviations are informative rather than defects. The P4 overrun discrepancy is the daily retirement-check granularity described above, which the closed form omits. The overproduction discrepancy arises because the closed-form estimate treats the shift capacity as deterministic, whereas cycle times are stochastic and the number of molds that are quota-limited rather than shift-limited varies from day to day; the simulation captures this interaction and the closed form cannot.

6) What the simulation contributes beyond the algebra. Relations (7) to (12) predict the mean of each response but not its dispersion, and they are silent on every interaction between mechanisms: between the counting rule and the retirement decision, between undercounting and defect generation, and between shift capacity and the interlock. They also cannot produce a confidence interval. The simulation supplies all of these, and the closed forms serve to verify that it is doing so correctly.

5.7 Implications for Deployment

Taken together, the results yield four requirements that the physical installation must satisfy.

- Middleware before hardware. The single-sided recovery layer of Section 3.4 must be implemented before the accuracy target can be met at any economically attainable detection probability. Specifying redundant antennas without it addresses the symptom rather than the cause.
- Detection probability floor. With the recovery layer, per-reader detection probability above 0.90 is sufficient; without it, 0.98 is required. Tag and antenna selection should be qualified against the lower figure.
- Gate calibration. The [25, 30] min window must be derived from the measured residence-time distribution of the specific oven, not adopted as a material constant.
- Oven process capability. Residence-time standard deviation must be held below approximately 1.0 min, which is a conveyor control requirement rather than an RFID requirement.

5.8 Threats to Validity

Five limitations qualify these results.

- No operational validation. The model reproduces plant-level defect rates and planning effort to within a few

percent, establishing face validity, but it has not been validated against measured RFID event streams. Reported accuracies are properties of the model, not measurements of an installed system.

- Input parameters are engineering estimates. Detection probabilities, manual error rates, and labour coefficients are assumed rather than calibrated. Table 6 bounds the consequence for the parameter of greatest uncertainty, but a corresponding calibration exercise is required for the remainder.
- Simplified failure modelling. Machine breakdown, operator absence, tag physical failure, and mold withdrawal for unscheduled maintenance are excluded. Including them would compress the differences between policies, since every policy degrades under disruption. The direction of the ranking is not expected to change, but the magnitude of the improvement is likely optimistic.
- Environmental channel not exercised. Oven temperature enters the model only implicitly, through the residence-time distribution. The root-cause tracing capability described in Section 3.5 is therefore part of the architecture but outside the scope of the present evaluation.
- Homogeneous line assumption. All ten lines share a single parameter set. In practice, antenna geometry and thermal exposure differ by line, so per-line detection probability would be heterogeneous.

6 Conclusion and Future Work

This study proposed a dual-tier RFID architecture for mold lifecycle and work-order execution control in polyurethane casting and evaluated it ex ante through stochastic discrete-event simulation of a ten-line, 120-mold shop.

The central result is a negative one about the architecture as first specified. Time-gated validation applied to strict two-reader pairing, the configuration most directly suggested by the literature on temporal filtering of RFID event streams, attains only 90.26% mold-level pass-count accuracy at realistic on-metal detection probabilities, below the 93.52% attained by the incumbent manual policy, and consequently increases expired-mold exposure. The mechanism is the one-directional nature of gate rejection. Single-sided detections are discarded, which produces a systematic undercount that the shop floor nevertheless treats as authoritative.

Augmenting the gate with a finite-state recovery layer resolves this. The resulting policy attains 98.06% (± 0.03) accuracy and remains above 96% even at a per-reader detection probability of 0.90. Over a 120-day horizon it reduces expired-mold exposure by 59.3%, defects

attributable to expired molds by 75.7%, overproduction by 78.9%, and production-planning labour by 90.4% relative to manual operation.

Sensitivity analysis converts these findings into three deployment requirements. Per-reader detection probability must exceed 0.90 with the recovery layer in place, against 0.98 without it. The curing gate must be calibrated to the measured residence-time distribution rather than fixed a priori. Oven residence-time standard deviation must be held below approximately 1.0 min.

6.1 Limitations

Three qualifications bound the claims made above; they are set out in full in Section 5.8. First, all reported performance figures are properties of the simulation model rather than measurements of an installed system: the model has been validated for structure and behaviour, and against plant-level defect and planning-effort records, but not for predictive accuracy against measured RFID event streams. Second, the physical-layer and manual-error parameters of Table 2 are engineering estimates rather than calibrated values, so the sensitivity bounds of Section 5.3 should be read as the operative result rather than the point estimates themselves. Third, the model excludes machine breakdown, tag failure and unscheduled mold withdrawal; since every policy degrades under such disruption, the ranking of the four policies is expected to hold but the magnitude of the improvement attributed to the recovery layer is likely optimistic. Until a pilot deployment supplies calibrated inputs, the contribution of this study is best read as a set of quantified design requirements rather than as a prediction of realised performance.

6.2 Future Work

Future work proceeds in three directions. First, calibration and operational validation: a pilot tag trial on the casting line will supply measured detection probabilities and residence-time distributions, permitting the model to be recalibrated and its predictions tested against measured event streams. Second, extension to an online digital twin, following the incremental DES-to-DT methodology of Wooley et al. [15], in which the validated offline model is connected to the live RFID stream to support real-time scheduling and predictive mold replacement. Third, relaxation of the uniform-wear assumption in the lifecycle tier: the present model treats every validated pass as an identical degradation unit ($\omega_i = 1$), whereas the synchronised oven-temperature channel described in Section 3.5 would permit a thermally weighted wear model and hence a more accurate remaining-useful-life estimate.

Acknowledgements

We acknowledge Ho Chi Minh City University of Technology (HCMUT), VNU-HCM for supporting this study.

References

1. Zhong RY, Dai QY, Qu T, Hu GJ, Huang GQ (2013) RFID-enabled real-time manufacturing execution system for mass-customization production. *Robot Comput Integr Manuf* 29(2):283-292. <https://doi.org/10.1016/j.rcim.2012.08.001>
2. de Jesus Pacheco DA, Jung CF, de Azambuja MC (2023) Towards industry 4.0 in practice: a novel RFID-based intelligent system for monitoring and optimisation of production systems. *J Intell Manuf* 34(3):1165-1181. <https://doi.org/10.1007/s10845-021-01840-w>
3. Dai Q, Zhong RY, Huang GQ, Qu T, Zhang T, Luo TY (2012) Radio frequency identification-enabled real-time manufacturing execution system: a case study in an automotive part manufacturer. *Int J Comput Integr Manuf* 25(1):51-65. <https://doi.org/10.1080/0951192X.2011.562546>
4. Fang J, Huang GQ, Li Z (2013) Event-driven multi-agent ubiquitous manufacturing execution platform for shop floor work-in-progress management. *Int J Prod Res* 51(4):1168-1185. <https://doi.org/10.1080/00207543.2012.693644>
5. Brintrup A, Ranasinghe D, McFarlane D (2010) RFID opportunity analysis for leaner manufacturing. *Int J Prod Res* 48(9):2745-2764. <https://doi.org/10.1080/00207540903156517>
6. Lu S, Xu C, Zhong RY, Wang L (2017) A RFID-enabled positioning system in automated guided vehicle for smart factories. *J Manuf Syst* 44:179-190. <https://doi.org/10.1016/j.jmsy.2017.03.009>
7. Zheng Y, Qiu S, Shen F, He C (2020) RFID-based material delivery method for mixed-model automobile assembly. *Comput Ind Eng* 139:106023. <https://doi.org/10.1016/j.cie.2019.106023>
8. Thirappa K, Lim EH, Bong FL, Chung BK (2021) Slim RFID tag antenna for metallic tools with narrow footprint. *IEEE J Radio Freq Identif* 5(2):182-190. <https://doi.org/10.1109/JRFID.2021.3057101>
9. Yeap HHL, Eu KS, Alexander CHC, Yap KM, Tan JKP (2024) Metal environmental effects on the performance of passive high frequency RFID tag readability. *AIP Conf Proc* 3161:020087. <https://doi.org/10.1063/5.0230066>
10. Seré A, Alsultan MA, Steinfeld L, Pérez-Nicoli P, Vilajosana X, Meliá-Seguí J (2024) Extending temperature range of UHF RFID sensors for industrial applications. *IEEE Sens Lett* 8(12):1-4. <https://doi.org/10.1109/LSSENS.2024.3506514>
11. Karagiannaki A, Doukidis G, Pramataris K (2014) A framework for mapping the RFID-enabled process redesign in a simulation model. *J Oper Res Soc* 65(11):1700-1710. <https://doi.org/10.1057/jors.2013.128>
12. Arkan I, Van Landeghem H (2013) Evaluating the performance of a discrete manufacturing process using RFID: a case study. *Robot Comput Integr Manuf* 29(6):502-512. <https://doi.org/10.1016/j.rcim.2013.06.003>
13. Florescu A (2024) Digital twin for flexible manufacturing systems and optimization through simulation: a case study. *Machines* 12(11):785. <https://doi.org/10.3390/machines12110785>
14. Battissacco BC, Azzolini Júnior W, dos Santos CH, Romano AL (2026) Discrete event simulation-based digital twin: an approach focused on tactical, strategic, and operational decisions. *J Simul* 20(3):307-325. <https://doi.org/10.1080/17477778.2025.2531045>

15. Wooley A, Bitencourt J, Silva DF (2025) Bridging the gap between discrete event simulation and digital twin: a manufacturing case study. *Manuf Lett* 44:1274-1284. <https://doi.org/10.1016/j.mfglet.2025.06.147>
16. Zhong RY, Huang GQ, Lan S, Dai QY, Zhang T, Xu C (2015) A two-level advanced production planning and scheduling model for RFID-enabled ubiquitous manufacturing. *Adv Eng Inform* 29(4):799-812. <https://doi.org/10.1016/j.aei.2015.01.002>
17. Barenji RV, Barenji AV, Hashemipour M (2014) A multi-agent RFID-enabled distributed control system for a flexible manufacturing shop. *Int J Adv Manuf Technol* 71(9-12):1773-1791. <https://doi.org/10.1007/s00170-013-5597-2>
18. Feng J, Li F, Xu C, Zhong RY (2020) Data-driven analysis for RFID-enabled smart factory: a case study. *IEEE Trans Syst Man Cybern Syst* 50(1):81-88. <https://doi.org/10.1109/TSMC.2018.2882838>
19. Ertek G, Chi X, Zhang AN (2017) A framework for mining RFID data from schedule-based systems. *IEEE Trans Syst Man Cybern Syst* 47(11):2967-2984. <https://doi.org/10.1109/TSMC.2016.2557762>
20. Palmgren A (1924) Die Lebensdauer von Kugellagern. *Z Ver Dtsch Ing* 68(14):339-341
21. Miner MA (1945) Cumulative damage in fatigue. *J Appl Mech* 12(3):A159-A164. <https://doi.org/10.1115/1.4009458>
22. Mejjaouli S, Babiceanu RF (2015) RFID-wireless sensor networks integration: decision models and optimization of logistics systems operations. *J Manuf Syst* 35:234-245. <https://doi.org/10.1016/j.jmsy.2015.02.005>
23. Rosenblatt MJ, Lee HL (1986) Economic production cycles with imperfect production processes. *IIE Trans* 18(1):48-55. <https://doi.org/10.1080/07408178608975329>
24. Kim CH, Hong Y (1999) An optimal production run length in deteriorating production processes. *Int J Prod Econ* 58(2):183-189. [https://doi.org/10.1016/S0925-5273\(98\)00119-4](https://doi.org/10.1016/S0925-5273(98)00119-4)
25. Hyndman RJ, Koehler AB (2006) Another look at measures of forecast accuracy. *Int J Forecast* 22(4):679-688. <https://doi.org/10.1016/j.ijforecast.2006.03.001>
26. Sargent RG (2013) Verification and validation of simulation models. *J Simul* 7(1):12-24. <https://doi.org/10.1057/jos.2012.20>

Ethics approval Not applicable. This study did not involve human participants or animals.

Use of large language models The use of a large language model in the preparation of this work is declared in full in Section 4.9.

Statements and Declarations

Funding The authors declare that no funds, grants or other support were received during the preparation of this manuscript.

Competing interests The authors have no other relevant financial or non-financial interests to disclose.

Author contributions The authors contributed to the study as follows:

- Quang Vinh Tran: Conceptualization, methodology, data collection, data analysis, visualization, and writing—original draft.
- Thi Diem Chau Le: Conceptualization, methodology, supervision, writing—review and editing.

Data availability The simulation model and the scripts that regenerate every table and figure reported in this article are available from the corresponding author on reasonable request.

Appendix A Discrete-Event Simulation Model

The listing below is the complete simulation model referred to throughout Section 4. PARAMS holds every input parameter of Table 2, POLICIES defines the four control policies of Table 1, and run_replication executes one replication of the production horizon under a given policy.

```
"""
pu_rfid_des.py
=====
Stochastic discrete-event simulation of a polyurethane (PU)
casting line
operating under four mold-tracking control policies.

Companion model for the paper:
"Real-Time Mold Lifecycle and Production Tracking System
Using RFID
Technology in Polyurethane Manufacturing"

Implementation
-----
Pure Python 3 + NumPy. No commercial simulation package is
required.
The model uses *common random numbers*: all physical-process
variates
(pouring, curing, de-mold, queueing, defect draws, reader
detections) are
drawn from a single primary stream that is consumed identically
under every
control policy, so policy comparisons are paired and variance is
reduced.
Policy-specific nuisance variates (manual batch-allocation
noise) are drawn
from an independent secondary stream so that they cannot
desynchronise the
primary stream.

Control policies
-----
MANUAL      paper tally per shift; mold life imputed from
batch yield
RFID_NAIVE  allocated equally over the molds of the line (as-
is baseline)
RFID_NAIVE  every Reader-Out detection increments P_ACTUAL (no
gate, no lock)
RFID_GATED  time-gated validation (25 <= dT <= 30) + capacity
interlock
RFID_GATED_R RFID_GATED + finite-state recovery of single-sided
reads

Response variables
-----
acc_mold    per-mold-day pass-count accuracy, 100*(1 - MAPE)
acc_line    per-line-day pass-count accuracy, 100*(1 - MAPE)
ret_err     |true passes at retirement - L_max| / L_max, %
life_wasted unused life at premature retirement, %
life_overrun passes beyond L_max at retirement, %
expired     mold-passes physically executed with true count >
L_max
overprod    bushes cast in excess of the daily work order
defect_rate visual-defect rate, %
def_expired defects occurring on expired-mold passes
hours_week  production-planning labour, h/week

Usage
-----
from pu_rfid_des import summarise, run_replication, PARAMS
print(summarise('RFID_GATED_R', n_reps=25))

P = dict(PARAMS); P['p_read_in'] = 0.90      # override any
parameter
print(summarise('RFID_GATED', n_reps=15, P=P))
"""
from __future__ import annotations
import numpy as np

__all__ = ['PARAMS', 'POLICIES', 'KEYS', 'run_replication',
'summarise',
'trajectory']

# -----
# Model parameters. Entries marked CALIBRATE must be replaced
with measured
# plant data before the results are used for anything beyond
design
# exploration.
# -----
```

```
PARAMS = dict(
    # -- shop configuration --
    n_lines=10,          # active casting lines
    molds_per_line=12,   # molds circulating per line
    horizon_days=20,     # production days per
replication
    shift_minutes=480,   # 8 h working shift
    L_max=600,           # structural mold lifespan,
passes
    # -- process times (minutes) --
    cure_mu=27.5, cure_sd=0.85, cure_lo=24.0, cure_hi=31.0, #
CALIBRATE
    pour_mu=2.4, pour_sd=0.5,
    demold_mu=3.1, demold_sd=0.7,
    queue_mu=4.6, queue_sd=1.8,

    # -- RFID physical layer --
    p_read_in=0.955,     # CALIBRATE from tag trial
    p_read_out=0.945,    # CALIBRATE from tag trial
    p_false_read=0.021,  # spurious / bounce read per
cycle

    # -- manual tally layer --
    p_manual_miss=0.062, # CALIBRATE from tally-vs-output
reconciliation
    p_manual_double=0.024, # CALIBRATE
    manual_shift_bias=0.045, # relative bias of the shift
batch total

    # -- quality model --
    p_def_base=0.0082,   # CALIBRATE from NG records
    def_wear_slope=0.00042, # CALIBRATE by regressing NG
rate on mold age

    # -- planning labour (hours) --
    h_per_record_manual=0.0075,
    h_shift_consolidation=0.78,
    h_per_exception=0.0333,
    h_daily_review=0.083,

    # -- validation gate --
    gate_lo=25.0, gate_hi=30.0,
    p_recover_in_only=0.72, # success rate of deferred
single-sided inference
    p_operator_overrun=0.38, # chance the operator casts one
extra before stopping
)

POLICIES = ('MANUAL', 'RFID_NAIVE', 'RFID_GATED',
'RFID_GATED_R')

KEYS = ('acc_mold', 'acc_line', 'ret_err', 'life_wasted',
'life_overrun',
'expired', 'overprod', 'defect_rate', 'def_expired',
'hours_week',
'true_total', 'false_acc', 'false_rej')

MAXC = 26 # upper bound on cycles a mold
can run in a shift

# -----
def run_replication(config, seed, P=PARAMS, track_mold=None):
    """Run one replication.

    Parameters
    -----
    config : str      one of POLICIES
    seed : int        seed for the primary (common) random
stream
    P : dict          parameter set
    track_mold : int | None if given, also return the daily
trajectory of
                    (true_cum, sys_cum) for that mold
index

    Returns
    -----
    dict of response variables; if track_mold is not None the
dict also
    carries 'traj_true' and 'traj_sys', each of length
horizon_days.
    """
    if config not in POLICIES:
        raise ValueError(f'unknown policy {config!r}; expected
one of {POLICIES}')

```

```

    rng = np.random.default_rng(seed)                # primary:
physical process                                     # double entry
    rng2 = np.random.default_rng(seed + 10_000_000)  #
secondary: manual noise

    n_lines, mpl = P['n_lines'], P['molds_per_line']
    nm = n_lines * mpl
    Lmax = P['L_max']
    line_of = np.repeat(np.arange(n_lines), mpl)

    mold_base = rng.integers(8, 18, nm)              #
persistent work-order level
    true_cum = rng.integers(0, Lmax, nm).astype(np.int64)
    sys_cum = true_cum.astype(float).copy()

    mold_ape, line_ape = [], []
    ret_err, life_wasted, life_ouerrun = [], [], []
    overprod = expired = defects = good = def_expired = 0
    false_acc = false_rej = tot_true = 0
    hours = 0.0
    traj_true, traj_sys = [], []

    for _ in range(P['horizon_days']):
        plan = np.clip(mold_base + rng.integers(-1, 2, nm), 6,
20)

        # pre-draw the day's variates (vectorised for speed)
        pour = np.clip(rng.normal(P['pour_mu'], P['pour_sd'],
(nm, MAXC)), 1.0, 6.0)
        cure = np.clip(rng.normal(P['cure_mu'], P['cure_sd'],
(nm, MAXC)),
                        P['cure_lo'], P['cure_hi'])
        demo = np.clip(rng.normal(P['demold_mu'],
P['demold_sd'], (nm, MAXC)), 1.0, 8.0)
        queue = np.clip(rng.normal(P['queue_mu'], P['queue_sd'],
(nm, MAXC)), 0.5, 15.0)
        cap = ((pour + cure + demo + queue).cumsum(axis=1) <=
P['shift_minutes']).sum(1)

        u_man = rng.random((nm, MAXC))
        r_in = rng.random((nm, MAXC)) < P['p_read_in']
        r_out = rng.random((nm, MAXC)) < P['p_read_out']
        r_fp = rng.random((nm, MAXC)) < P['p_false_read']
        r_rec = rng.random((nm, MAXC)) < P['p_recover_in_only']
        r_ovr = rng.random((nm, MAXC))
        u_def = rng.random((nm, MAXC))
        in_gate = (cure >= P['gate_lo']) & (cure <=
P['gate_hi'])

        d_true = np.zeros(nm, int)
        d_count = np.zeros(nm, float)
        n_records = n_exc = 0

        for m in range(nm):
            p_plan = int(plan[m])
            actual = phys = 0
            for c in range(int(cap[m])):
                if actual >= p_plan:
                    if config in ('RFID_GATED', 'RFID_GATED_R'):
                        break
# hard interlock
                if r_ovr[m, c] > P['p_operator_ouerrun']:
                    break
# operator stops

                # ----- physical reality -----
                phys += 1
                true_cum[m] += 1
                over = true_cum[m] - Lmax
                if over > 0:
                    expired += 1
                    p_def = min(0.35, P['p_def_base'] +
P['def_wear_slope'] * over)
                else:
                    p_def = P['p_def_base']
                if u_def[m, c] < p_def:
                    defects += 1
                    if over > 0:
                        def_expired += 1
                else:
                    good += 1

                # ----- observation layer -----
                if config == 'MANUAL':
                    u = u_man[m, c]
                    if u < P['p_manual_miss']:
                        pass
# omitted
                    elif u < P['p_manual_miss'] +
P['p_manual_double']:

    rng = np.random.default_rng(seed)                # primary:
physical process                                     # double entry
    rng2 = np.random.default_rng(seed + 10_000_000)  #
secondary: manual noise

    n_lines, mpl = P['n_lines'], P['molds_per_line']
    nm = n_lines * mpl
    Lmax = P['L_max']
    line_of = np.repeat(np.arange(n_lines), mpl)

    mold_base = rng.integers(8, 18, nm)              #
persistent work-order level
    true_cum = rng.integers(0, Lmax, nm).astype(np.int64)
    sys_cum = true_cum.astype(float).copy()

    mold_ape, line_ape = [], []
    ret_err, life_wasted, life_ouerrun = [], [], []
    overprod = expired = defects = good = def_expired = 0
    false_acc = false_rej = tot_true = 0
    hours = 0.0
    traj_true, traj_sys = [], []

    for _ in range(P['horizon_days']):
        plan = np.clip(mold_base + rng.integers(-1, 2, nm), 6,
20)

        # pre-draw the day's variates (vectorised for speed)
        pour = np.clip(rng.normal(P['pour_mu'], P['pour_sd'],
(nm, MAXC)), 1.0, 6.0)
        cure = np.clip(rng.normal(P['cure_mu'], P['cure_sd'],
(nm, MAXC)),
                        P['cure_lo'], P['cure_hi'])
        demo = np.clip(rng.normal(P['demold_mu'],
P['demold_sd'], (nm, MAXC)), 1.0, 8.0)
        queue = np.clip(rng.normal(P['queue_mu'], P['queue_sd'],
(nm, MAXC)), 0.5, 15.0)
        cap = ((pour + cure + demo + queue).cumsum(axis=1) <=
P['shift_minutes']).sum(1)

        u_man = rng.random((nm, MAXC))
        r_in = rng.random((nm, MAXC)) < P['p_read_in']
        r_out = rng.random((nm, MAXC)) < P['p_read_out']
        r_fp = rng.random((nm, MAXC)) < P['p_false_read']
        r_rec = rng.random((nm, MAXC)) < P['p_recover_in_only']
        r_ovr = rng.random((nm, MAXC))
        u_def = rng.random((nm, MAXC))
        in_gate = (cure >= P['gate_lo']) & (cure <=
P['gate_hi'])

        d_true = np.zeros(nm, int)
        d_count = np.zeros(nm, float)
        n_records = n_exc = 0

        for m in range(nm):
            p_plan = int(plan[m])
            actual = phys = 0
            for c in range(int(cap[m])):
                if actual >= p_plan:
                    if config in ('RFID_GATED', 'RFID_GATED_R'):
                        break
# hard interlock
                if r_ovr[m, c] > P['p_operator_ouerrun']:
                    break
# operator stops

                # ----- physical reality -----
                phys += 1
                true_cum[m] += 1
                over = true_cum[m] - Lmax
                if over > 0:
                    expired += 1
                    p_def = min(0.35, P['p_def_base'] +
P['def_wear_slope'] * over)
                else:
                    p_def = P['p_def_base']
                if u_def[m, c] < p_def:
                    defects += 1
                    if over > 0:
                        def_expired += 1
                else:
                    good += 1

                # ----- observation layer -----
                if config == 'MANUAL':
                    u = u_man[m, c]
                    if u < P['p_manual_miss']:
                        pass
# omitted
                    elif u < P['p_manual_miss'] +
P['p_manual_double']:

    actual += 2
    else:
        actual += 1
        n_records += 1

    elif config == 'RFID_NAIVE':
        inc = int(r_out[m, c]) + int(r_fp[m, c])
        if r_fp[m, c]:
            false_acc += 1
        actual += inc

    elif config == 'RFID_GATED':
        if r_in[m, c] and r_out[m, c] and in_gate[m,
c]:
            actual += 1
        else:
            n_exc += 1
            false_rej += 1

    else: # RFID_GATED_R
        ok = ((r_in[m, c] and r_out[m, c] and
in_gate[m, c])
              or (r_out[m, c] and not r_in[m, c] and
in_gate[m, c])
              or (r_in[m, c] and not r_out[m, c] and
r_rec[m, c]))
        if ok:
            actual += 1
        else:
            n_exc += 1
            false_rej += 1

        d_true[m] = phys
        d_count[m] = actual
        if phys > p_plan:
            overprod += (phys - p_plan)

        tot_true += int(d_true.sum())

        # ---- how the system updates each mold's cumulative
        life counter ----
        if config == 'MANUAL':
            # end-of-shift batch yield of the line allocated
            equally over its
            # molds: the source of the persistent per-mold bias
            for l in range(n_lines):
                idx = np.where(line_of == l)[0]
                batch = d_count[idx].sum() * (1.0 +
rng2.normal(0, P['manual_shift_bias']))
                sys_cum[idx] += batch / len(idx)
            else:
                sys_cum += d_count

        ok = d_true > 0
        mold_ape.extend(np.abs(d_count[ok] - d_true[ok]) /
d_true[ok])
        for l in range(n_lines):
            idx = np.where(line_of == l)[0]
            t, c = d_true[idx].sum(), d_count[idx].sum()
            if t > 0:
                line_ape.append(abs(c - t) / t)

        if track_mold is not None:
            traj_true.append(int(true_cum[track_mold]))
            traj_sys.append(float(sys_cum[track_mold]))

        # ---- retirement decision ----
        ret = sys_cum >= Lmax
        if ret.any():
            d = (true_cum[ret] - Lmax) / Lmax
            ret_err.extend(np.abs(d))
            life_wasted.extend(d[d < 0])
            life_ouerrun.extend(d[d > 0])
            true_cum[ret] = 0
            sys_cum[ret] = 0.0

        # ---- planning labour ----
        if config == 'MANUAL':
            hours += (n_records * P['h_per_record_manual']
+ n_lines * P['h_shift_consolidation'])
        else:
            hours += (n_exc * P['h_per_exception']
+ n_lines * P['h_daily_review'])

        out = dict(
            acc_mold=100.0 * (1.0 - float(np.mean(mold_ape))),
            acc_line=100.0 * (1.0 - float(np.mean(line_ape))),
            ret_err=100.0 * float(np.mean(ret_err)) if ret_err else
0.0,
            life_wasted=100.0 * float(np.mean(life_wasted)) if
life_wasted else 0.0,

```

```

        life_ouerrun=100.0 * float(np.mean(life_ouerrun)) if
life_ouerrun else 0.0,
        expired=expired,
        overprod=overprod,
        defect_rate=100.0 * defects / max(1, defects + good),
        def_expired=def_expired,
        hours_week=hours / P['horizon_days'] * 5,
        true_total=tot_true,
        false_acc=false_acc,
        false_rej=false_rej,
    )
    if track_mold is not None:
        out['traj_true'] = traj_true
        out['traj_sys'] = traj_sys
    return out

def summarise(config, n_reps=25, base_seed=20260101, P=PARAMS):
    """Mean and 95% confidence half-width of every response over
n_reps runs."""
    rows = [run_replication(config, base_seed + r, P) for r in
range(n_reps)]
    return {k: (float(np.mean([r[k] for r in rows])),
                1.96 * float(np.std([r[k] for r in rows],
ddof=1)) / np.sqrt(n_reps))
            for k in KEYS}

def trajectory(config, mold=0, seed=20260101, P=PARAMS):
    """Daily (true, system) cumulative pass counts for one
mold."""
    r = run_replication(config, seed, P, track_mold=mold)
    return np.array(r['traj_true']), np.array(r['traj_sys'],
dtype=float)

if __name__ == '__main__':
    print(f"{'Policy':<14}{'AccMold%':>10}{'Expired':>10}{'Overprod'
:>10}{'h/week':>10}")
    for pol in POLICIES:
        s = summarise(pol, n_reps=25)

    print(f'{pol:<14}{s["acc_mold"][0]:>10.2f}{s["expired"][0]:>10.0
f}')

    f'{s["overprod"][0]:>10.0f}{s["hours_week"][0]:>10.1f}')

```

Appendix B Experiment Driver

The listing below regenerates every numerical result and every figure reported in Section 5. The run is deterministic: the model is seeded at 20260101 and all four policies consume a single common random-number stream in identical order.

```
"""
run_experiments.py
=====
Regenerates every numerical result and every figure reported in
the paper.
```

```
python3 run_experiments.py
```

Outputs

```
-----
tables/table3_baseline.csv      policy comparison, 20-day
horizon
tables/table4_extended.csv      policy comparison, 120-day
horizon
tables/table5_readrate.csv      sensitivity to reader
detection probability
tables/table6_gatewidth.csv     sensitivity to validation gate
width
tables/table7_ovenvar.csv       sensitivity to oven residence-
time variability
figures/fig4_policy_accuracy.png bar chart with 95% CI
figures/fig5_readrate.png      accuracy vs detection
probability
figures/fig6_ovenvar.png        accuracy vs oven variability
figures/fig7_trajectory.png     true vs system pass count for
one mold
```

Figures follow IEEE formatting guidance: single-column width (3.5 in), 300 dpi, serif type, 8 pt labels, axis quantities spelled out with units in parentheses.

```
"""
from __future__ import annotations
import os
import csv
import json
import sys
import numpy as np
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
from scipy.stats import norm
```

```
from pu_rfid_des import PARAMS, POLICIES, summarise,
run_replication, trajectory
```

```
os.makedirs('tables', exist_ok=True)
os.makedirs('figures', exist_ok=True)
```

```
CACHE = 'results_cache.json'
FIGS_ONLY = '--figures-only' in sys.argv
if FIGS_ONLY and not os.path.exists(CACHE):
    raise SystemExit('no results_cache.json; run without --
figures-only first')
```

```
plt.rcParams.update({
    'font.family': 'serif',
    'font.size': 8,
    'axes.labelsize': 8,
    'axes.titlesize': 8,
    'legend.fontsize': 7,
    'xtick.labelsize': 7,
    'ytick.labelsize': 7,
    'axes.grid': True,
    'grid.alpha': 0.3,
    'grid.linewidth': 0.4,
    'lines.linewidth': 1.2,
    'figure.dpi': 300,
})
W1, W2 = 3.5, 7.16          # IEEE single- and double-column
widths, inches
LBL = {'MANUAL': 'P1 Manual', 'RFID_NAIVE': 'P2 RFID naive',
       'RFID_GATED': 'P3 RFID gated', 'RFID_GATED_R': 'P4 RFID
gated + recovery'}
```

```
def write_csv(path, header, rows):
    with open(path, 'w', newline='') as f:
        w = csv.writer(f)
        w.writerow(header)
        w.writerows(rows)
    print(f' wrote {path}')
```

```
def fmt(t):
    return f'{t[0]:.2f} +/- {t[1]:.2f}'
```

```
if FIGS_ONLY:
    _c = json.load(open(CACHE))
    base = {k: {kk: tuple(vv) for kk, vv in v.items()} for k, v
in _c['base'].items()}
    ext = {k: {kk: tuple(vv) for kk, vv in v.items()} for k, v
in _c['ext'].items()}
    ps = _c['ps']; gated = [tuple(x) for x in _c['gated']]
    recov = [tuple(x) for x in _c['recov']]
    sds = _c['sds']; acc_sd = [tuple(x) for x in _c['acc_sd']]
    share_sd = _c['share_sd']
    print('loaded cached results; regenerating figures only')
```

```
# -----
-- TABLE III
if not FIGS_ONLY:
    print('\nTABLE III Policy comparison, 20 production days, 25
replications')
    base = {}
    rows = []
    for p in POLICIES:
        s = summarise(p, n_reps=25)
        base[p] = s
        rows.append([LBL[p], fmt(s['acc_mold']),
fmt(s['acc_line']), fmt(s['ret_err']),
fmt(s['expired']), fmt(s['overprod']),
fmt(s['hours_week']),
f'{s['life_overrun']}[0]:.2f}"],
f'{s['life_overrun']}[0]:.2f}"]])
    print(f' {LBL[p]:<28} acc={s["acc_mold"][0]:6.2f}
expired={s["expired"][0]:6.0f}
f' overprod={s["overprod"][0]:6.0f}
h/wk={s["hours_week"][0]:5.1f}')
    write_csv('tables/table3_baseline.csv',
[Policy', 'Mold accuracy (%)', 'Line accuracy (%)',
'Retirement error (%)',
'Expired passes', 'Overproduced bushes', 'Planning
labour (h/week)',
'Wasted life (%)', 'Overrun life (%)'], rows)
```

```
# -----
-- TABLE IV
print('\nTABLE IV Policy comparison, 120 production days, 15
replications')
PL = dict(PARAMS); PL['horizon_days'] = 120
ext = {}
rows = []
for p in POLICIES:
    s = summarise(p, n_reps=15, P=PL)
    ext[p] = s
    rows.append([LBL[p], fmt(s['acc_mold']),
fmt(s['expired']), fmt(s['def_expired']),
f'{s["defect_rate"][0]:.3f} +/-
{s["defect_rate"][1]:.3f}'],
fmt(s['overprod'])])
    print(f' {LBL[p]:<28} acc={s["acc_mold"][0]:6.2f}
expired={s["expired"][0]:7.0f}
f' def_exp={s["def_expired"][0]:6.1f}
NG%={s["defect_rate"][0]:.3f}')
    write_csv('tables/table4_extended.csv',
[Policy', 'Mold accuracy (%)', 'Expired passes',
'Defects on expired molds',
'Defect rate (%)', 'Overproduced bushes'], rows)
```

```
b, g = ext['MANUAL'], ext['RFID_GATED_R']
print('\n P4 versus P1 over 120 days:')
for k, lab in (('expired', 'expired-mold passes'),
('def_expired', 'defects on expired molds'),
('defect_rate', 'overall defect rate'),
('overprod', 'overproduced bushes'),
('hours_week', 'planning labour')):
    print(f' {lab:<28} (b[k][0]:9.2f) -> (g[k][0]:9.2f)
f' ({100*(g[k][0]-b[k][0])/b[k][0]:+6.1f} %))')
    print(f' {"mold-level accuracy":<28}
{b["acc_mold"][0]:9.2f} -> {g["acc_mold"][0]:9.2f}
f' ({g["acc_mold"][0]-b["acc_mold"][0]:+6.2f} pp)')
    print(f'\n Face validity: manual NG on an 18,299-bush order =
f'{18299*b["defect_rate"][0]/100:.1f} (dashboard
reference: 159)')
```

```
# -----
-- TABLE V
print('\nTABLE V Sensitivity to reader detection
probability')
ps = [0.90, 0.92, 0.94, 0.955, 0.97, 0.99]
gated, recov = [], []
rows = []
```

```

for p in ps:
    P1 = dict(PARAMS); P1['p_read_in'] = p; P1['p_read_out'] =
p - 0.01
    a = summarise('RFID_GATED', n_reps=15, P=P1)['acc_mold']
    c = summarise('RFID_GATED_R', n_reps=15, P=P1)['acc_mold']
    gated.append(a); recov.append(c)
    rows.append([f'{p:.3f}/{p-0.01:.3f}', fmt(a), fmt(c)])
    print(f' p={p:.3f}/{p-0.01:.3f} P3={a[0]:6.2f}
P4={c[0]:6.2f}')
    write_csv('tables/table5_readrate.csv',
        ['p_in / p_out', 'P3 gated accuracy (%)', 'P4
gated+recovery accuracy (%)'], rows)

# -----
---- TABLE VI
print('\nTABLE VI Sensitivity to validation gate width')
rows = []
for lo, hi in [(26.0, 29.0), (25.5, 29.5), (25.0, 30.0),
(24.0, 31.0), (23.0, 32.0)]:
    P1 = dict(PARAMS); P1['gate_lo'] = lo; P1['gate_hi'] = hi
    s = summarise('RFID_GATED_R', n_reps=15, P=P1)
    rows.append([f'{lo:.1f}', {hi:.1f}], ffmt(s['acc_mold']),
        f'{s['false_rej']}[0]:.0f}',
f'{s['hours_week']}[0]:.1f}'))
    print(f' gate [{lo:.1f},{hi:.1f}]
acc={s['acc_mold']}[0]:6.2f}'
        f' rejected={s['false_rej']}[0]:6.0f}
h/wk={s['hours_week']}[0]:5.1f}')
    write_csv('tables/table6_gatewidth.csv',
        ['Gate (min)', 'Mold accuracy (%)', 'Rejected
reads', 'Planning labour (h/week)'],
        rows)

# -----
---- TABLE VII
print('\nTABLE VII Sensitivity to oven residence-time
variability')
sds = [0.60, 0.85, 1.20, 1.60, 2.20]
acc_sd, share_sd = [], []
rows = []
for sd in sds:
    P1 = dict(PARAMS); P1['cure_sd'] = sd
    share = 100 * (norm.cdf((P1['gate_hi'] - P1['cure_mu']) /
sd)
        - norm.cdf((P1['gate_lo'] - P1['cure_mu'])
/ sd))
    s = summarise('RFID_GATED_R', n_reps=15, P=P1)['acc_mold']
    acc_sd.append(s); share_sd.append(share)
    rows.append([f'{sd:.2f}', f'{share:.2f}', ffmt(s)])
    print(f' sd={sd:.2f} min in-gate={share:6.2f} %
acc={s[0]:6.2f}')
    write_csv('tables/table7_ovenvar.csv',
        ['Oven residence-time SD (min)', 'Passes inside gate
(%)',
        'Mold accuracy (%)'], rows)

    json.dump(dict(base=base, ext=ext, ps=ps, gated=gated,
recov=recov,
        sds=sds, acc_sd=acc_sd, share_sd=share_sd),
        open(CACHE, 'w'), indent=1)
    print(f' wrote {CACHE}')

# -----
-- plotting
INK, GRID, HL = '#1a1a1a', '#d9d9d9', '#8c8c8c'

def frame(ax):
    """Two-sided axis frame, ticks outward, horizontal reference
lines only."""
    for side in ('top', 'right'):
        ax.spines[side].set_visible(False)
    for side in ('left', 'bottom'):
        ax.spines[side].set_linewidth(0.7)
        ax.spines[side].set_color(INK)
    ax.tick_params(direction='out', length=2.6, width=0.6,
colors=INK, pad=2)
    ax.grid(True, axis='y', color=GRID, linewidth=0.5)
    ax.set_axisbelow(True)

# -----
---- FIG. 4
fig, ax = plt.subplots(figsize=(W1, 2.35))
vals = [base[p]['acc_mold'] for p in POLICIES]
errs = [base[p]['acc_mold'][1] for p in POLICIES]
shade = ['c9c9c9', 'c9c9c9', '8f8f8f', '3a3a3a']
ax.bar(range(4), vals, yerr=errs, capsize=2.5, width=0.62,
color=shade,
        edgecolor=INK, linewidth=0.6,
        error_kw=dict(elinewidth=0.7, capthick=0.7, ecolor=INK))
ax.axhline(95, ls=(0, (4, 3)), c=HL, lw=0.8, zorder=1)

```

```

ax.text(3.45, 95.4, 'target', ha='right', fontsize=6.5,
color=HL)
ax.set_xticks(range(4))
ax.set_xticklabels(['P1\manual', 'P2\nnaive', 'P3\ngated',
'P4\ngated +\nrecovery'])
ax.set_ylabel('Pass-count accuracy (percent)')
ax.set_ylim(86, 100.5)
ax.set_yticks([86, 90, 94, 98])
for i, v in enumerate(vals):
    ax.text(i, v + 0.6, f'{v:.2f}', ha='center', fontsize=6.6,
color=INK)
frame(ax)
fig.tight_layout()
fig.savefig('figures/fig4_policy_accuracy.png',
bbox_inches='tight', facecolor='white')
plt.close(fig)
print('\n wrote figures/fig4_policy_accuracy.png')

# -----
---- FIG. 5
fig, ax = plt.subplots(figsize=(W1, 2.45))
ax.axhline(95, c=HL, ls=(0, (4, 3)), lw=0.8, zorder=1)
ax.errorbar(ps, [a[0] for a in gated], yerr=[a[1] for a in
gated],
        marker='s', ms=3.2, color=INK, ls='-', lw=1.0,
        capsize=1.8,
        elinewidth=0.6, label='P3 gate only')
ax.errorbar(ps, [a[0] for a in recov], yerr=[a[1] for a in
recov],
        marker='o', ms=3.4, color=INK, ls=(0, (4, 2)),
        lw=1.0,
        mfc='white', mew=0.9, capsize=1.8, elinewidth=0.6,
        label='P4 gate + recovery')
ax.text(0.9015, 94.3, 'accuracy target', fontsize=6.5, color=HL,
va='top')
ax.set_xlabel('Reader detection probability')
ax.set_ylabel('Pass-count accuracy (percent)')
ax.set_xlim(0.893, 0.997)
ax.set_ylim(78, 101)
ax.set_xticks([0.90, 0.92, 0.94, 0.96, 0.98])
ax.set_yticks([80, 85, 90, 95, 100])
ax.legend(loc='lower right', frameon=False, handlelength=2.4,
borderpad=0.2)
frame(ax)
fig.tight_layout()
fig.savefig('figures/fig5_readrate.png', bbox_inches='tight',
facecolor='white')
plt.close(fig)
print(' wrote figures/fig5_readrate.png')

# -----
---- FIG. 6
fig, ax = plt.subplots(figsize=(W1, 2.45))
ax.axhline(95, c=HL, ls=(0, (4, 3)), lw=0.8, zorder=1)
ax.axvspan(0.45, 1.0, color='#f0f0f0', zorder=0)
ax.errorbar(sds, [a[0] for a in acc_sd], yerr=[a[1] for a in
acc_sd],
        marker='o', ms=3.4, color=INK, lw=1.0, capsize=1.8,
        elinewidth=0.6,
        label='Pass-count accuracy')
ax.plot(sds, share_sd, marker='^', ms=3.2, color=HL, ls=(0, (4,
2)), lw=1.0,
        label='Passes inside gate')
ax.text(0.72, 107, 'acceptable dispersion', fontsize=6.5,
color=HL,
        ha='center', va='top')
ax.set_xlabel('Oven residence-time standard deviation
(minutes)')
ax.set_ylabel('Percent')
ax.set_xlim(0.45, 2.35)
ax.set_ylim(70, 108)
ax.set_yticks([70, 80, 90, 100])
ax.set_xticks([0.5, 1.0, 1.5, 2.0])
ax.legend(loc='lower left', frameon=False, handlelength=2.4,
borderpad=0.2,
        bbox_to_anchor=(0.30, 0.02))
frame(ax)
fig.tight_layout()
fig.savefig('figures/fig6_ovenvar.png', bbox_inches='tight',
facecolor='white')
plt.close(fig)
print(' wrote figures/fig6_ovenvar.png')

# -----
---- FIG. 7
PT = dict(PARAMS); PT['horizon_days'] = 90
rng = np.random.default_rng(20260101)
mold_base = rng.integers(8, 18, PARAMS['n_lines'] *
PARAMS['molds_per_line'])
target = int(np.argmax(mold_base))

fig, axes = plt.subplots(1, 3, figsize=(W2, 2.15), sharey=True)
for k, (ax, pol) in enumerate(zip(axes, ['MANUAL', 'RFID_GATED',
'RFID_GATED_R'])):
    tt, ss = trajectory(pol, mold=target, seed=20260101, P=PT)
    d = np.arange(1, len(tt) + 1)

```

```

    ax.axhline(600, c=HL, ls=(0, (4, 3)), lw=0.8, zorder=1)
    ax.plot(d, tt, color=INK, ls='-', lw=1.0, label='Actual
passes')
    ax.plot(d, ss, color=HL, ls=(0, (4, 2)), lw=1.0,
label='System count')
    ax.set_title(LBL[pol], fontsize=7.6, pad=4)
    ax.set_xlabel('Production day')
    ax.set_xlim(0, 92)
    ax.set_ylim(0, 760)
    ax.set_xticks([0, 30, 60, 90])
    frame(ax)
    if k:
        ax.spines['left'].set_visible(False)
        ax.tick_params(left=False)
axes[0].set_ylabel('Cumulative passes')
axes[0].set_yticks([0, 200, 400, 600])
axes[0].text(3, 700, 'structural limit, 600 passes',
fontsize=6.4, color=HL)
h, l = axes[0].get_legend_handles_labels()
fig.legend(h, l, loc='lower center', ncol=2, frameon=False,
fontsize=7,
            handlelength=2.4, bbox_to_anchor=(0.5, -0.06))
fig.tight_layout(w_pad=1.0)
fig.savefig('figures/fig7_trajectory.png', bbox_inches='tight',
facecolor='white')
plt.close(fig)
print(f' wrote figures/fig7_trajectory.png (mold #{target}, '
      f'work order {mold_base[target]} passes/day)')

print('\nDone.')

```

Supplementary Files

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

- [Dataset2.txt](#)
- [Dataset.txt](#)