SOCIAL AND MANAGEMENT SCIENCES RESEARCH JOURNAL

www.smsrj.com

p-ISSN: 2791-0415 **e-ISSN:** 2791-0423

Volume:4 Issue: 02(2026)

Which Delay Matters? A Cause-Aware Waiting-Time Decomposition for Internal-Control Risk Detection in Business Process Event Logs

Yu-Ting Lin, Chia-Hao Huang, Wei-Jen Chen*

Department of Computer Science and Information Engineering, National Central University, Taoyuan, Taiwan

**Corresponding author: Wei-Jen Chen, weijen.chen1@ncu.edu.tw*

Abstract:

Operational delay and internal-control risk are normally analysed by different functions with different tools, and the assumption that accelerating a process also improves control is rarely tested on real data. This paper asks whether process delay carries information about internal-control risk and, if so, which component of delay carries it. We introduce a pool-level, capacity-aware decomposition of pre-activity waiting time into resource contention, calendar unavailability and residual delay that is exactly computable on the kind of log real finance functions actually own: one timestamp per event and anonymised resources. The contention window of a waiting interval is obtained in closed form from a k-th-largest departure rule, giving an O(n log k) algorithm. We combine the decomposition with a prefix-safe observation design in which every item is observed at its first control point, so that all features precede the control decision that defines the label. On three public event logs (two BPI Challenge 2020 expense-approval logs and the CoSeLoG receipt-phase log; 18,497 analysed items) we find that waiting is dominated by calendar unavailability (75.3-76.1%), that total waiting is nevertheless a robust item-level risk indicator (odds ratio per standard deviation 1.27, 1.09 and 1.46 after quarter, role, amount and off-hours controls), and that ranking items by waiting time alone retrieves 21.3%, 17.3% and 24.2% of all control exceptions within a 10% inspection budget. Conditional on total waiting, the working-time component carries the risk in two of the three logs while calendar-idle waiting does not. Two negative results qualify the practice: the incremental AUC of delay features over context and conformance features is small (+0.011 to +0.015), and the delay-risk association exists at item level but disappears at period level once secular trend is removed. Delay is therefore a cheap and reliable signal for audit sample selection, not a period-level control indicator.

Keywords: *process mining; internal control; audit analytics; waiting time analysis; queue mining; predictive process monitoring; risk assessment*

1. Introduction

Administrative and financial approval processes — expense reimbursement, payment requests, journal-entry approval, permit issuance — are executed thousands of times per year against fixed deadlines and finite pools of human reviewers. Two questions dominate their management. The controller asks where the process loses time. The internal auditor asks which items carry a material risk that a control did not operate as intended. In practice the two questions are answered by different functions, from different data extracts, with different tools, and the answers are rarely reconciled. The separation is uncomfortable because the two phenomena are widely believed to interact. Congestion at a shared approval pool pushes work towards the deadline, which is exactly when reviews are compressed, approvals are batched and items are handled outside business hours; conversely, control failures generate rework, which lengthens the cycle. A recent framework for the multi-entity financial close makes this intuition operational: it runs a performance track and an internal-control track over one event log and joins them through an explicit coupling analysis, reporting a strong delay-risk coupling in a controlled setting together with the useful observation that the coupling weakens where waiting is idle calendar time rather than contention [1]. That result is valuable and motivates the present study, but it also raises the question that we address here: on real logs with observable control outcomes, does the coupling survive, and which part of the delay is responsible for it? Answering the question empirically is harder than it appears, for three reasons. First, waiting time is not a homogeneous quantity. An activity that accumulates a long pre-activity gap because the responsible pool is saturated is operationally different from one whose gap merely spans a weekend, yet both look identical to any aggregate waiting-time metric. Second, the logs that finance functions actually own are impoverished: they typically record a single timestamp per event and, after anonymisation, a role rather than an individual resource, which rules out the resource-level busy-interval computations used in the queue-mining literature. Third, and most damaging for empirical work, control outcomes are recorded as events inside the same trace, so a case-level encoding that uses the whole trace leaks the outcome into the features and produces optimistic results that cannot be reproduced in operation.

This paper addresses all three issues and reports what the data support. Our contributions are as follows.

1) *A cause-aware waiting-time decomposition for impoverished logs.* We decompose every pre-activity gap into resource contention, calendar unavailability and residual delay at the level of the resource pool that executes the activity, so that role-anonymised logs are still analysable. We show that the contention part of a waiting interval is an initial sub-interval whose right endpoint is the k-th largest departure time among earlier arrivals in the same pool, which yields an exact O(n log k) algorithm rather than an approximation (Section III-B).

2) *A prefix-safe control-risk observation design.* Every item is observed at its first control point; all features are measured strictly before the control decision and the label is the outcome of that decision or of the remaining suffix. The design removes the leakage that a whole-trace encoding introduces and matches the moment at which an auditor or a controller would actually intervene (Section III-C).

3) *A multi-log empirical study of the delay-risk link.* We analyse 18,497 items from three public logs with objective, log-derivable control outcomes, and report association, ranking power,

incremental predictive value, an audit-sampling experiment and four robustness analyses (Sections IV and V).

4) *Two negative results that qualify practice.* The incremental discrimination that delay features add on top of item context and conformance features is small and its confidence interval includes zero in all three logs; and the association that is robust at item level vanishes at period level once the secular trend is removed, so weekly cycle-time monitoring is not a substitute for item-level risk scoring.

II. RELATED WORK

A. Process mining for auditing and internal control

The use of event logs as audit evidence was proposed early [2] and demonstrated in a field study on procurement, which showed that process mining reveals violations of segregation of duties and authorisation limits that sampling-based procedures miss [3], [4]. Subsequent work formalised rule-based compliance checking with risk management [5], diagnosed where deviations occur [6], and studied the accounting data structures from which financial event logs must be inferred [7], multilevel analysis for financial audits [8], the detection of absent internal controls [9], the reduction of false positives in fraud detection [10], the evaluation of internal-control effectiveness [11] and the embedding of process mining into statutory audits [12]. Conformance checking supplies the formal machinery for this line of work [13]-[15]. The literature is predominantly rule-based and conformance-oriented: it asks whether a prescribed control was executed, not whether the operational state of the process makes a control failure more likely. The quantitative link between congestion and control risk is therefore largely unexamined on real data, which is the gap this paper addresses.

B. Waiting time, queueing and performance mining

Performance mining traditionally aggregates sojourn or waiting times per activity or per directly-follows edge [16], [17]. Because aggregation hides variability, the performance spectrum was introduced to describe performance without aggregation and to analyse batching [18], [19]. Queue mining models the delay caused by shared resources explicitly and predicts delays from queueing information [20], [21], and inference techniques recover unobserved queueing events in systems with shared resources [22]. A recent line of work classifies the direct causes of waiting time — batching, prioritisation, contention, resource unavailability and extraneous factors — and quantifies their contribution to cycle-time efficiency [23], [24]; extraneous delays have also been modelled explicitly to improve simulation fidelity [25]. Our decomposition follows this causal taxonomy but departs from it in one respect that matters for financial logs: because individual resources are anonymised, contention is computed at the level of the pool with an estimated habitual load, which makes the decomposition applicable where resource-level busy intervals cannot be reconstructed.

C. Outcome-oriented predictive process monitoring

Predicting a case-level outcome from a running prefix is a mature field with established benchmarks, in which gradient boosting on tabular case encodings remains a strong baseline against deep models [26]-[28]. Post-hoc explanation has been transferred to this setting [29]-[31], and prescriptive extensions optimise interventions for cycle-time reduction [32]. We adopt the prefix-based protocol of this literature, but the object of prediction is different: rather than a business outcome such as a deadline or a cancellation, the label is the operation of an internal control, and the feature of interest is the causal composition of the delay that the item experienced before the control was executed.

III. METHOD

A. Event log, pools and waiting intervals

An event log is a set of events $e = (c, a, r, t)$ where c is the case (here: one item, such as one declaration), a the activity, r the executing resource and t the timestamp. We assume a single timestamp per event, which is the common case in financial systems, and we write $p(e)$ for the resource pool responsible for e, taken as the organisational role or group recorded on the event. For the i-th event of case c we define the waiting interval as $[a_i, b_i)$ where b_i is the timestamp of the event and a_i the timestamp of its predecessor in the case; operationally, a_i is the moment the item enters the pool $p(e_i)$ and b_i the moment the pool acts on it. All quantities below are defined per waiting interval and aggregated per item.

B. Cause-aware decomposition of waiting time

We split the gap $g = b - a$ of every waiting interval into three additive parts,

$$g = g_{con} + g_{res} + g_{una},$$

where g_{una} is the part of g that falls outside the working calendar, g_{con} is the part inside the working calendar during which the pool is congested, and g_{res} is the remaining working-calendar time, which absorbs batching, hand-over and extraneous delay. The working calendar is Monday to Friday, 09:00-17:00 local time; the calendar part is obtained from a monotone business-time transform $B(t)$ that returns elapsed working seconds since a fixed origin, so that the working part of a gap is exactly $B(b) - B(a)$ and $g_{una} = (b - a) - (B(b) - B(a))$.

The congested part requires a definition of congestion that survives resource anonymisation. Let the waiting intervals assigned to a pool p be $I_j = [a_j, b_j)$. At time t the number of items that entered p before j and are still in p is $n_j(t) = |\{i : a_i < a_j, b_i > t\}|$. We call the item congested while $n_j(t)$ is at least k_p, where k_p is the habitual load of the pool, estimated as the median queue length observed on arrival. The following observation makes the computation exact and cheap.

Proposition 1. Let M_j be the k-th largest element of $\{b_i : a_i < a_j\}$. Then $\{t \in [a_j, b_j) : n_j(t) \geq k\} = [a_j, \min(b_j, M_j))$.

The proof is immediate: $n_j(t)$ counts earlier arrivals whose departure time exceeds t, and since an item that has left the pool never returns, n_j is non-increasing in t; therefore $n_j(t) \geq k$ holds exactly while t is smaller than the k-th largest departure time among earlier arrivals. Consequently the congested window of every waiting interval is an initial sub-interval whose endpoint is available from a single sweep of the pool in arrival order, maintaining the k largest departure times in a min-heap; the whole decomposition costs $O(n \log k)$ time and $O(k)$ memory per pool. We then set $g_{con} = B(\min(b_j, M_j)) - B(a_j)$, so that the congested part is measured in working time and the three components add up to g by construction. Setting $k = 1$ recovers the strict reading of resource contention — the pool is occupied with earlier work — while larger k measures excess congestion relative to the load the pool normally carries; we report both as a sensitivity analysis. Besides the three components we retain two state variables that an analyst can read off the log at the moment an item enters the queue: the queue length on arrival $n_j(a_j)$ and the pool backlog, i.e. the number of items waiting for the pool at that time. We also compute the batch pressure of a decision, defined as the number of other events executed by the same pool within one hour of it, as a proxy for compressed handling.

C. Prefix-safe observation of control risk

Control outcomes are themselves events. Encoding the whole trace therefore leaks the label: the number of events, the rework ratio and the hand-over count of a rejected item all reflect the rejection they are supposed to predict. We avoid this by observing every item once, at its first control point, defined as the first event executed by a role other than the submitting one. The

prefix consists of all events up to and including that decision; every feature is computed on the prefix and the label is the outcome of the decision (for the expense logs) or of the remaining suffix (for the deadline log). Because the waiting that precedes a decision is complete before the decision is taken, the delay features are available at the moment of use in both an ex-ante and an ex-post reading: the state variables of Section III-B are known when the item enters the queue, and the realised decomposition is known immediately before the control is executed, which is precisely when an auditor selecting items for testing would apply it. Four blocks of features are formed. Block X (context) contains the item amount, the hour, weekday and off-hours flag of submission. Block C (control conformance) contains the resubmission and save flags, the hour, weekday and off-hours flag of the decision and the batch pressure. Block A (aggregate delay) contains the total and the maximum waiting time of the prefix. Block D (decomposed delay) contains the three components, their shares, the queue length on arrival and the pool backlog. The contrast between A and D isolates the value of the cause decomposition; the contrast between X+C and X+C+A+D isolates the value of delay information as such.

D. Evaluation protocol

We evaluate three complementary questions. Association: logistic regression of the control outcome on standardised delay terms with fixed effects for calendar quarter and for the deciding role and controls for item amount and off-hours handling; odds ratios are reported per standard deviation with 95% percentile confidence intervals from 300 bootstrap resamples, and an effect is called significant when its interval excludes one. Ranking: the area under the ROC curve of each single indicator used directly as a score, with 500 bootstrap resamples, together with the lift obtained at inspection budgets of 10% and 20%, which is the quantity an audit planner uses. Prediction: L2 logistic regression and histogram gradient boosting trained on the first 70% of items in chronological order and tested on the last 30%, with the difference in test AUC between nested feature sets assessed by a paired bootstrap on the test split. All protocols are strictly out-of-time; no information from the test period is used at any stage.

IV. EXPERIMENTAL SETUP

A. Event logs and control outcomes

No public financial-close log exists, so we use three public logs that share the properties the method relies on: finite shared human pools, an explicit approval or checking step, and an objective, log-derivable outcome that reflects the operation of a control.

Domestic Declarations (DD) and Requests for Payment (RFP) are two logs of the BPI Challenge 2020 collection [33], recording two years of travel-expense and payment-request handling at a Dutch university. An item is submitted by an employee and then checked by the travel administration, the budget owner, the pre-approver and the supervisor; the role of the actor is recorded, the individual is not. The control outcome is a rejection pronounced by an authority role, which is the observable event in which the approval control detects a defect in the item. We label an item positive if such a rejection is pronounced at its first control point or later.

Receipt phase (RCP) is the receipt phase of an environmental permit application process of a Dutch municipality, collected in the CoSeLoG project [34]. It carries a statutory deadline per case, which yields an objective compliance outcome; it records individual resources and organisational groups, and it contains check/determine pairs that implement a four-eyes control. We use it to test whether the findings transfer to a different domain and to a deadline-based rather than rejection-based outcome, observing every case after its third event.

Table I summarises the logs. The three settings differ in size, in domain and in outcome semantics, but agree closely in the composition of their waiting time, which is dominated by calendar unavailability.

Table 1Characteristics of the three event logs and composition of waiting time

Property	**DD**	**RFP**	**RCP**
Cases in the log	10,500	6,886	1,434
Events	56,437	36,796	8,577
Activities	17	19	27
Resource pools	7	8	10
Analysed items	10,365	6,814	1,318
Control-exception rate (%)	9.33	12.31	5.39
Median cycle time (days)	7.33	8.15	0.03
Mean waiting per step (h)	63.2	66.3	26.1
Contention (% of waiting)	3.4	3.8	9.2
Calendar (% of waiting)	75.3	75.4	76.1
Residual (% of waiting)	21.2	20.8	14.7

Contention is computed with the habitual-load rule k_p = median queue length on arrival. DD = Domestic Declarations, RFP = Requests for Payment, RCP = Receipt phase.

B. Implementation

The pipeline is implemented in Python 3.12 with pandas, numpy, scipy, statsmodels and scikit-learn [35]; XES logs are parsed with a dedicated streaming reader, and results were cross-checked against pm4py conventions for log statistics [36]. Gradient boosting uses histogram-based trees with depth 3, 250 iterations and learning rate 0.06; the logistic models use L2 regularisation. All code, the derived feature tables and every result file are released with the paper; the logs themselves are public.

V. RESULTS

A. Where the waiting time goes

Waiting dominates flow time in all three processes, and its causal composition is remarkably stable across them: 75.3%, 75.4% and 76.1% of all waiting is calendar unavailability, that is, time in which no one was expected to work (Fig. 1). Under the strict contention rule (k = 1, i.e. at least one earlier item still in the pool) 22.1%, 19.5% and 19.1% of waiting is contention, which is almost the whole of the working-time portion; under the habitual-load rule the excess-congestion component shrinks to 3.4%, 3.8% and 9.2% and the remainder is attributed to the residual. The practical reading is that in these processes an item that waits a long time mostly waits because of the calendar, and only a minority of the waiting reflects a queue that is longer than the pool normally carries.

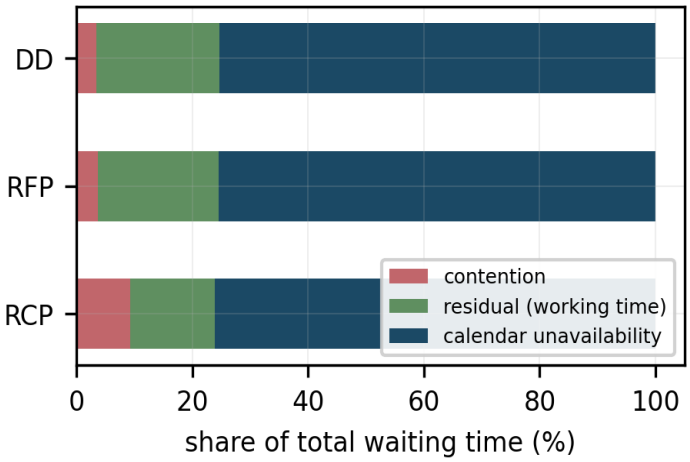


Figure 1Causal composition of waiting time in the three logs. Calendar unavailability accounts for three quarters of all waiting in every process.

The distribution of the waiting that precedes the first control point is strongly bimodal. In DD the median item is handled 50 seconds after submission, the 90th percentile waits 0.7 hours and the maximum waits 469 days; items that are eventually rejected by the administration wait on average 152 hours against 31 hours for items that are approved. Most items therefore pass through the first control instantaneously, and the minority that queues is the interesting one — which is exactly what makes waiting time a usable triage signal.

B. Delay as a risk-ranking signal

Table II reports the discrimination of each indicator used directly as a score, with no model and no fitting. Total waiting is the strongest single indicator in DD and RFP (AUC 0.583 and 0.535) and is essentially tied with the residual and calendar components in RCP (0.580 against 0.594 and 0.589); the 95% bootstrap intervals for total waiting are [0.564, 0.602], [0.513, 0.555] and [0.504, 0.660], so the effect is separated from chance in all three logs. Discrimination of this magnitude is modest in absolute terms, but it is obtained from a single number that requires no labels, no training and no model maintenance, and it translates into a useful lift: inspecting the 10% of items with the longest waiting time yields 1.67, 1.51 and 2.27 times the exception density of random inspection.

Table 2Risk-ranking power of single delay indicators (AUC and lift at a 10% inspection budget)

Indicator	**DD AUC**	**L@10**	**RFP AUC**	**L@10**	**RCP AUC**	**L@10**
Total waiting	0.583	1.67	0.535	1.51	0.580	2.27
Working-time waiting	0.556	1.69	0.520	1.43	0.563	2.41
Contention	0.546	1.75	0.543	1.41	0.512	2.13
Residual (working time)	0.509	1.61	0.472	1.10	0.594	2.41
Calendar unavailability	0.534	1.65	0.523	1.31	0.589	1.98
Queue length on arrival	0.555	1.26	0.570	1.32	0.442	0.28

Indicator	DD AUC	L@10	RFP AUC	L@10	RCP AUC	L@10
Batch pressure at decision	0.550	1.56	0.518	1.16	0.400	1.13

L@10 = lift at a 10% inspection budget. AUC values are computed on all analysed items; 95% bootstrap intervals are given in the text.

Fig. 2 shows why the aggregate indicator works and where it stops working. In all three logs the exception rate is flat across the first four quintiles of waiting and jumps in the fifth: from 8.3% to 15.4% in DD, from 10.3% to 16.1% in RFP and from 3.0% to 9.9% in RCP. The relationship is therefore not monotone but threshold-like, which is consistent with the bimodal waiting distribution and which explains why linear models extract less from delay than tree-based ones.

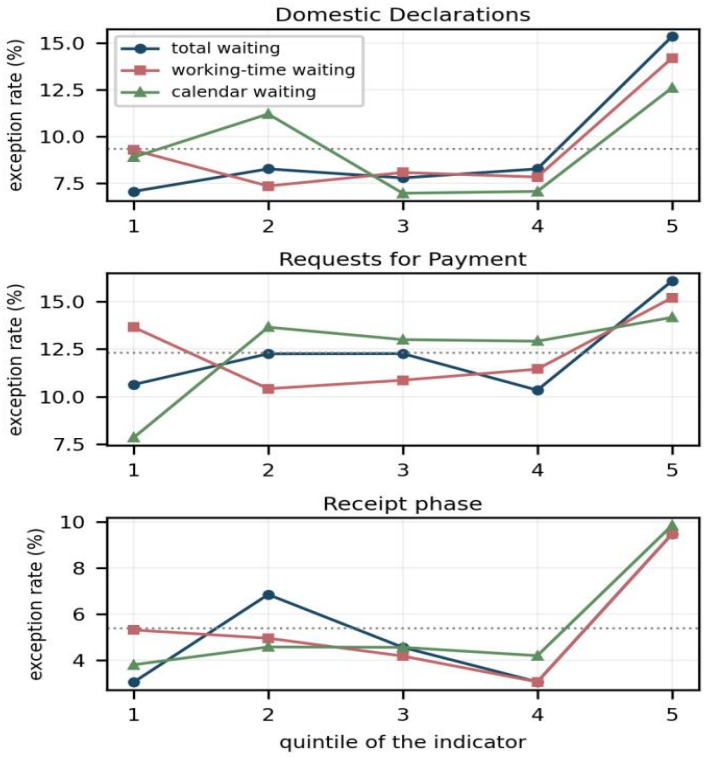


Figure 2Control-exception rate by quintile of three delay indicators. The dotted line is the base rate. The risk is concentrated in the top quintile in all three logs.

C. Association and the role of composition

Table III reports odds ratios per standard deviation from logistic models with quarter and role fixed effects and controls for amount and off-hours handling. Total waiting is positively associated with a control exception in all three logs, with odds ratios of 1.269, 1.094 and 1.456 and intervals that exclude one; the association is therefore not an artefact of the secular growth of the process or of the mix of deciding roles, since both are absorbed by the fixed effects.

Model M2 asks a sharper question: holding the amount of waiting fixed, does its composition matter? In DD and RCP it clearly does. Shifting waiting from calendar-idle time to working time raises the odds of an exception substantially (OR 2.544 and 3.170), while the calendar component is negatively associated (0.488 and 0.446). The full decomposition M3 attributes the effect in DD to both working-time components (contention 1.097, residual 1.377) and in RCP

mainly to the residual (1.966) with contention positive as well (1.284). RFP reverses the pattern: there the calendar component is the positively associated one (1.412) and the working-time component is not significant. We report this disagreement as a boundary condition rather than smoothing it over: the direction of the composition effect is process-specific, whereas the association of total waiting is not.

Table 3Odds ratios per standard deviation of delay, with quarter and role fixed effects

Model / term	**DD**	**RFP**	**RCP**
M1 total waiting	1.269*	1.094*	1.456*
95% CI	[1.21, 1.33]	[1.03, 1.16]	[1.18, 1.70]
M2 working-time waiting	2.544*	0.768	3.170*
95% CI	[1.73, 3.66]	[0.56, 1.07]	[1.60, 6.55]
M2 calendar waiting	0.488*	1.412*	0.446*
95% CI	[0.34, 0.74]	[1.01, 1.93]	[0.21, 0.90]
M3 contention	1.097*	1.005	1.284*
95% CI	[1.02, 1.19]	[0.88, 1.11]	[1.01, 1.88]
M3 residual	1.377*	0.897	1.966*
95% CI	[1.11, 1.80]	[0.76, 1.07]	[1.42, 3.42]
M3 calendar	0.858	1.192	0.615*
95% CI	[0.64, 1.08]	[0.97, 1.49]	[0.27, 0.96]
Items	10,365	6,814	1,318

* the 95% bootstrap interval excludes 1. All models include fixed effects for calendar quarter and deciding role and controls for item amount, off-hours submission and off-hours decision.

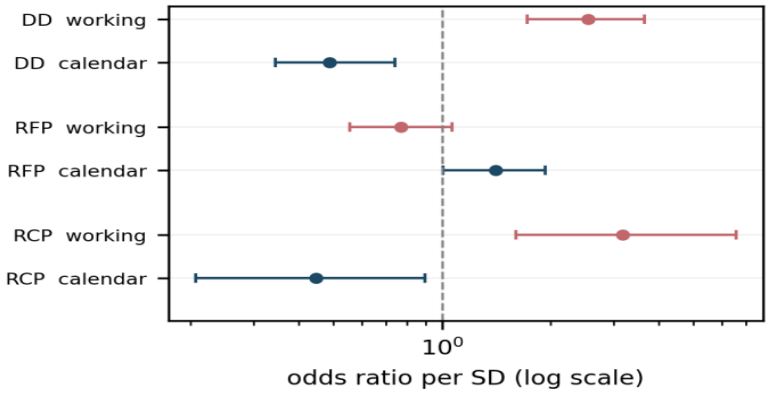


Figure 3Composition effect (model M2). In two of three logs the working-time component of waiting raises exception odds while calendar-idle waiting does not.

D. Incremental predictive value

Table IV evaluates the same information inside a predictive model under an out-of-time split. Two observations matter. First, delay is informative but far from sufficient: a model built on delay features alone reaches 0.568, 0.542 and 0.501 AUC, which is close to what the single best indicator already achieves. Second, once item context and conformance features are present, the increment is small. The paired bootstrap gives an AUC difference of +0.011 [−0.007, 0.031] in DD, +0.015 [−0.022, 0.052] in RFP and −0.005 [−0.127, 0.104] in RCP for adding all delay blocks, and the cause decomposition adds nothing beyond aggregate delay for prediction (−0.004, +0.001 and −0.053). We consider the honest reading to be that delay information is a complement of small marginal size in a well-specified model, not a substitute for item-level attributes. The reason is visible in the DD log: the item amount alone reaches 0.743 AUC under a gradient-boosted model because its relation to the exception rate is strongly U-shaped — 26.0% of the smallest decile of amounts and 16.8% of the largest are rejected, against 3.8-5.4% in the middle deciles. Where such a strong item-level predictor exists, delay can add little; where it does not, as in RCP, the aggregate-delay model is the best of the four (0.643 AUC against 0.596 without delay).

Table 4Out-of-time predictive performance by feature set

Log	Feature set	LR	GBM	AP	Lift@20
DD	X context	0.560	0.749	0.444	2.79
	X+C conformance	0.571	0.743	0.436	2.62
	X+C+A aggregate	0.599	0.757	0.454	2.84
	X+C+A+D decomposed	0.605	0.753	0.451	2.83
	delay only (A+D)	0.557	0.568	0.171	1.62
RFP	X context	0.567	0.582	0.171	1.48
	X+C conformance	0.564	0.561	0.208	1.44
	X+C+A aggregate	0.569	0.574	0.208	1.56
	X+C+A+D decomposed	0.572	0.575	0.170	1.57
	delay only (A+D)	0.516	0.542	0.156	1.18
RCP	X context	0.525	0.469	0.094	0.91
	X+C conformance	0.630	0.596	0.119	1.37
	X+C+A aggregate	0.637	0.643	0.178	1.82
	X+C+A+D decomposed	0.633	0.589	0.119	1.52

Log	Feature set	LR	GBM	AP	Lift@20
	delay only (A+D)	0.567	0.501	0.088	1.06

LR and GBM are test-set AUC; AP is average precision of the GBM. Training on the first 70% of items in chronological order, testing on the last 30%.

E. Audit-sampling experiment

For an auditor the operative quantity is not AUC but how many exceptions a fixed inspection budget retrieves. Table V evaluates selection rules on the out-of-time test period. Ranking by waiting time alone retrieves 21.3%, 17.3% and 24.2% of all exceptions within a 10% budget, against 9.2%, 9.4% and 12.1% for random selection and 16.2% and 15.4% for ranking by item amount, the rule most audit functions use today. In DD the full model retrieves 41.9% at a 10% budget and 56.5% at a 20% budget; adding delay to the context-and-conformance model raises the 20% coverage from 52.4% to 56.5% and in RFP from 28.7% to 31.5%, that is, four and three additional percentage points of the exception population at no additional cost. In RCP the model-free waiting rule outperforms both models, which is consistent with the small sample available for training there.

Table 5Share of all control exceptions retrieved by inspection budget

Log	Selection rule	@10%	Lift	@20%	Lift
DD	random	9.2	0.92	17.1	0.86
	item amount	16.2	1.62	27.0	1.35
	total waiting	21.3	2.13	39.7	1.98
	queue on arrival	12.1	1.21	22.5	1.13
	model X+C	41.3	4.13	52.4	2.62
	model X+C+delay	41.9	4.19	56.5	2.83
RFP	random	9.4	0.95	18.9	0.94
	item amount	15.4	1.54	29.9	1.50
	total waiting	17.3	1.74	28.0	1.40
	queue on arrival	13.4	1.34	22.4	1.12
	model X+C	17.3	1.74	28.7	1.44
	model X+C+delay	16.5	1.66	31.5	1.57
RCP	random	12.1	1.23	30.3	1.52
	total waiting	24.2	2.46	36.4	1.82
	queue on arrival	24.2	2.46	24.2	1.22
	model X+C	18.2	1.85	27.3	1.37
	model X+C+delay	15.2	1.54	30.3	1.52

Values are the percentage of all exceptions in the test period that fall in the inspected sample. Evaluated on the out-of-time test split.

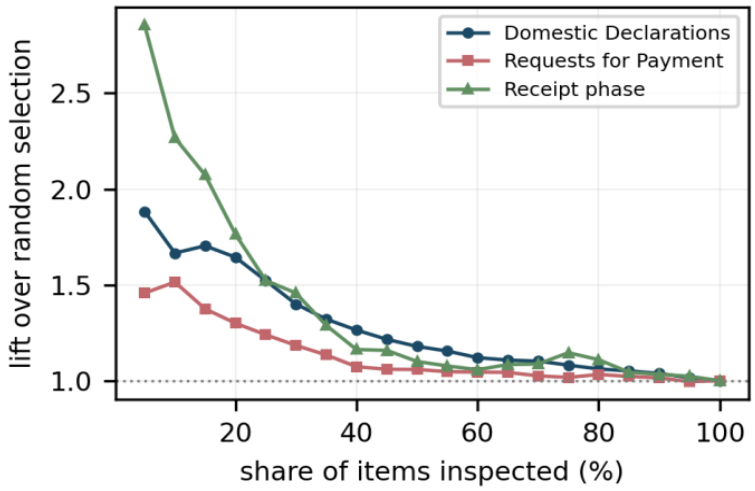


Figure 4Lift of a waiting-time-based inspection rule as a function of the inspection budget, computed on all analysed items

F. Robustness and mechanism

Capacity rule. Table VI varies the habitual-load parameter k_p. The direction of the contention effect is stable in DD and RCP across all three rules (odds ratios 2.263 and 2.015 under the strict rule, still above one under the conservative ones) and stable in its disagreement in RFP. What changes is the share of waiting that the rule calls contention, from about 20% at k = 1 to a few percent at the 75th-percentile rule, which is a reminder that any statement of the form "x% of waiting is contention" is a statement about the capacity assumption as much as about the process.

Table 6Sensitivity of the decomposition and of the contention effect to the habitual-load rule

Log	Rule	Contention %	OR con.	OR res.	OR cal.
DD	k = 1	22.1	2.263	1.051	0.544
	k = median	1.8	1.097	1.377	0.858
	k = p75	0.6	1.065	1.671	0.726
RFP	k = 1	19.5	0.707	0.950	1.538
	k = median	3.1	1.005	0.897	1.192
	k = p75	1.7	0.976	0.845	1.285
RCP	k = 1	19.1	2.015	1.343	0.614
	k = median	7.0	1.284	1.966	0.615
	k = p75	3.8	1.068	2.365	0.573

Contention % is the share of total waiting classified as contention. Odds ratios come from model M3 with the same controls as Table III.

Item level versus period level. If congestion caused control failures at the level of the process, weeks with more congestion should show higher exception rates. They appear to: the weekly correlation between mean queue length and the weekly exception rate is 0.377 in DD and 0.446

in RFP (both $p < 0.001$). The correlation is however entirely a trend artefact. Removing the year mean from both series drives it to −0.025 ($p = 0.81$) in DD, 0.122 ($p = 0.30$) in RFP and 0.114 ($p = 0.54$) in RCP, and the weekly correlation of mean waiting time with the exception rate is indistinguishable from zero even before detrending (0.023, 0.015 and −0.026). The delay-risk link that survives fixed effects at item level therefore does not exist at period level in these processes. Aggregate cycle-time monitoring will not reveal it, and a close-acceleration programme should not be expected to reduce the exception rate as a by-product.

What is the association made of? Table VII reports two diagnostic tests. Replacing the item-specific queue length by the expected congestion of its arrival slot — the leave-one-out mean queue length of the role, weekday, time-of-day and month cell, which is the part of congestion that is predictable from the calendar alone and cannot be caused by the item itself — yields odds ratios of 0.958, 1.175 and 1.309 with intervals that all include one. The risk signal is therefore carried by the waiting that is specific to the item, not by systematic congestion of the slot in which it arrived, which points at case-specific handling (difficult or doubtful items being deferred) rather than at pure queueing pressure. Consistently, in DD items that pass the first control after a long wait remain more likely to be rejected later (OR 1.198, interval [1.06, 1.30]), so the signal survives the control it precedes; in RFP the same test is not significant (0.958).

Table 7Diagnostic tests on the source of the delay signal

Log	Test	OR	95% CI	n
DD	expected congestion of slot	0.958	[0.73, 1.27]	9,515
RFP	expected congestion of slot	1.175	[0.97, 1.36]	5,934
RCP	expected congestion of slot	1.309	[0.69, 2.31]	759
DD	later rejection after approval	1.198	[1.06, 1.30]	9,398
RFP	later rejection after approval	0.958	[0.78, 1.07]	5,975

The first three rows replace item-specific congestion by the congestion expected in the arrival slot. The last two rows model a later rejection among items approved at the first control point, as a function of total waiting before that approval.

VI. DISCUSSION

A. Implications

Three results are directly actionable. First, waiting time is a cheap and reliable input to audit sample selection. It needs no labels, no training and no model governance, and in the three processes studied it retrieves between 1.5 and 2.5 times more exceptions than random selection at a 10% budget and clearly outperforms selection by item amount, which remains the dominant practice. Where a trained model is available, adding delay features raises the coverage of a 20% inspection budget by three to four percentage points of the exception population.

Second, the composition of waiting should be inspected before delay is interpreted. Three quarters of the waiting in these processes is calendar-idle time, and in two of the three logs that component carries no risk signal at all once total waiting is held fixed, while working-time waiting does. An activity that merely sits before a weekend is not a control-risk indicator; a queue that forms during working hours may be. The decomposition costs one sweep of the log and can

be computed before any risk model is trained, which makes it a natural screening step — an operational refinement of the observation, made in the multi-entity close setting, that delay and control risk are coupled only where waiting is contention-driven [1]. Third, the level of analysis matters more than the literature suggests. The same data support a robust item-level association and no period-level association whatsoever. Management dashboards that track average cycle time per week and infer control health from it are reading a signal that our data do not contain; the signal lives in the dispersion across items, not in the period mean.

B. Threats to validity

The control outcome is a proxy. A rejection pronounced by an authority role is evidence that a control detected a defect, not evidence of an undetected control failure, and the deadline breach used in RCP is a compliance rather than an audit outcome. Items whose defects were never detected are unobservable in any log, which bounds what can be concluded from any observational study of this kind; our escape analysis addresses one part of this gap but not all of it. The association is not causal. We control for calendar quarter, deciding role, item amount and off-hours handling, and we show that the signal is item-specific rather than slot-specific, but reverse causality remains plausible and, in our reading, likely: an approver who is unsure about an item may defer it, so that the item both waits longer and is more likely to be rejected. This does not weaken the practical claim — the indicator is available before the decision and is therefore usable for triage — but it does weaken any claim that removing the delay would remove the risk. The pool-level contention estimator depends on an assumed capacity, and the anonymisation of resources in two of the three logs prevents the resource-level computation that the waiting-time taxonomy prescribes; the sensitivity analysis of Table VI quantifies how much of the reported composition follows from that assumption. Finally, three logs from two domains and two countries are a narrow empirical base, and the disagreement of RFP with the other two logs shows that the direction of the composition effect does not generalise automatically.

C. Future work

Three directions follow. The first is to obtain logs in which individual resources are visible, so that the pool-level estimator can be validated against resource-level busy intervals and so that approver fixed effects can separate deferral behaviour from queueing pressure. The second is to move from association to intervention, using causal estimators on event logs to ask what a capacity change would do to the exception rate rather than what a longer wait predicts. The third is object-centric: expense claims, permits and payments are naturally objects with their own life cycles, and flattening them into one case notion discards the inter-object congestion that our pool-level estimator only approximates.

VII. CONCLUSION

We asked whether process delay carries information about internal-control risk and which part of the delay carries it. Using an exact, capacity-aware decomposition of waiting time that works on single-timestamp, role-anonymised logs, and a prefix-safe design that observes every item at its first control point, we showed on three public logs and 18,497 items that total waiting is a robust item-level risk indicator (odds ratios of 1.27, 1.09 and 1.46 per standard deviation after quarter, role and amount controls) and a useful audit-sampling signal (21.3%, 17.3% and 24.2% of all exceptions retrieved within a 10% budget). Conditional on the amount of waiting, its composition matters in two of the three logs, where working-time waiting is associated with control exceptions and calendar-idle waiting is not. We also reported what the data do not support: the incremental discrimination of delay features over item context and conformance features is small, and the association that survives fixed effects at item level disappears at period

level once trend is removed. Delay is therefore best used as an item-level triage signal, decomposed before it is trusted, and not as a period-level indicator of control health.

REFERENCES

Zhang, S. and Qiu, L. (2024). Data-driven assessment of pavement performance and resilience under extreme weather using the Long-Term Pavement Performance Program. AI and Data Science Journal, vol. 1, no. 1, pp. 57–66

Zhang, H. (2024). Graph-Aware Multi-Task Learning For Early Prediction Of Timing Violations And Routing Congestion In Digital Ic Physical Design. World Journal of Information Technology, 2(1): 13-20.

Wang, J., Tan, Y., Jiang, B., Wu, B., & Liu, W. (2025). Dynamic marketing uplift modeling: A symmetry-preserving framework integrating causal forests with deep reinforcement learning for personalized intervention strategies. Symmetry, 17(4), 610.

Zhao, X., Liu, J., Wang, Y., & Wang, J. (2026). CryptoMamba-SSM: Linear complexity state space models for cryptocurrency volatility prediction. IEEE Open Journal of the Computer Society, 7, 226-243.

Wang, J., Liu, J., Zheng, W., & Ge, Y. (2025). Temporal heterogeneous graph contrastive learning for fraud detection in credit card transactions. IEEE Access.

Sun, T., Yang, J., Li, J., Chen, J., Liu, M., Fan, L., & Wang, X. (2024). Enhancing auto insurance risk evaluation with transformer and SHAP. IEEE Access, 12, 116546-116557.

Zhang, H. (2025). Reinforcement Learning Approaches for Layout Optimization in Electronic Design Automation with Electromagnetic Compatibility Constraints. Frontiers in Robotics and Automation, 2(2), 77-93.

Han, X., Yang, Y., Chen, J., Wang, M., & Zhou, M. (2025). Symmetry-aware credit risk modeling: A deep learning framework exploiting financial data balance and invariance. Symmetry, 17(3), 341.

Chen, J., Cui, Y., Zhang, X., Yang, J., & Zhou, M. (2024). Temporal convolutional network for carbon tax projection: A data-driven approach. Applied Sciences, 14(20), 9213.

Sun, T., Wang, M., & Chen, J. (2025). Leveraging machine learning for tax fraud detection and risk scoring in corporate filings. Asian Business Research Journal, 10(11), 1-13.

Chen, J., Wang, M., & Sun, T. (2025). Intelligent tax systems and the role of natural language processing in regulatory interpretation. American Journal of Machine Learning, 6(4), 74-94.

Chen, Z., Liu, J., & Chen, J. (2025). Machine Learning Methods for Financial Forecasting in Enterprise Planning: Transitioning from Rule-Based Models to Predictive Analytics. Frontiers in Artificial Intelligence Research, 2(3), 541-564.

Chen, J., Liu, J., Liang, Y., & Zhou, M. (2026). KE-MLLM: A knowledge-enhanced multi-sensor learning framework for explainable fake review detection. Applied Sciences, 16(6), 2909.

Yue, X., Yang, J., & Liu, W. (2026). FraudDebate-Agent: A Multi-Agent LLM Framework with an Evidence-Based Debate Mechanism for Financial Statement Fraud Detection. Mathematics, 14(15), 2695.

Zhu, R. & Yue, X. (2024). An Explainable Machine Learning Framework For Predicting Dividend Sustainability And Financial Risk Of U.s. Reits Under Interest-Rate Shocks. Journal of Trends in Finance and Economics, 1(2): 48-57.

Yue, X. & Zhu, R. (2024). An Explainable Hybrid Machine Learning Framework For Cash-Flow-Conditioned Multi-Horizon Liquidity Risk Early Warning: An Sme-Oriented Benchmark Study. World Journal of Management Science, 2(1): 63-72.

Cui, Y., Han, X., Chen, J., Zhang, X., Yang, J., & Zhang, X. (2025). FraudGNN-RL: a graph neural network with reinforcement learning for adaptive financial fraud detection. IEEE Open Journal of the Computer Society, 6, 426-437.

Zi, B. (2024). Cloud-native distributed systems for real-time payment intelligence. AI and Data Science Journal, 1(1), 51-56.

Zhang, S. & Qiu, L. (2023). An Interpretable, Uncertainty-Aware Framework For Bridge Deterioration Prediction And Risk-Based Maintenance Prioritization. Academic Journal of Architecture and Civil Engineering, 1(2): 21-28.

Jin, J., Xing, S., Ji, E., & Liu, W. (2025). Xgate: explainable reinforcement learning for transparent and trustworthy api traffic management in iot sensor networks. Sensors, 25(7), 2183.

Xing, S., Wang, Y., & Liu, W. (2025). Multi-dimensional anomaly detection and fault localization in microservice architectures: A dual-channel deep learning approach with causal inference for intelligent sensing. Sensors, 25(11), 3396.

Xing, S., & Wang, Y. (2025). Proactive data placement in heterogeneous storage systems via predictive multi-objective reinforcement learning. IEEE Access, 13, 117986-117998.

Ji, E., Wang, Y., Xing, S., & Jin, J. (2025). Hierarchical reinforcement learning for energy-efficient API traffic optimization in large-scale advertising systems. IEEE Access.

Xing, S., & Wang, Y. (2025). Cross-modal attention networks for multi-modal anomaly detection in system software. IEEE Open Journal of the Computer Society.

Liu, Y., Ren, S., Wang, X., & Zhou, M. (2024). Temporal logical attention network for log-based anomaly detection in distributed systems. Sensors, 24(24), 7949.

Ren, S., Jin, J., Niu, G., & Liu, Y. (2025). ARCS: Adaptive reinforcement learning framework

for automated cybersecurity incident response strategy optimization. Applied Sciences, 15(2), 951.

Li, P., Ren, S., Zhang, Q., Wang, X., & Liu, Y. (2024). Think4SCND: Reinforcement learning with thinking model for dynamic supply chain network design. IEEE Access, 12, 195974-195985.

Hu, X., Zhao, X., Wang, J., & Yang, Y. (2025). Information-theoretic multi-scale geometric pre-training for enhanced molecular property prediction. Plos one, 20(10), e0332640.

Wang, M., Zhang, X., Yang, Y., & Wang, J. (2025). Explainable machine learning in risk management: Balancing accuracy and interpretability. Journal of Financial Risk Management, 14(3), 185-198.

Yang, J., Li, P., Cui, Y., Han, X., & Zhou, M. (2025). Multi-sensor temporal fusion transformer for stock performance prediction: An adaptive Sharpe ratio approach. Sensors, 25(3), 976.

Zhao, X., Sun, T., Ren, S., Yang, J., & Liu, Y. (2025). RAG-Based AI Agents for Enterprise Software Development: Implementation Patterns and Production Deployment. Frontiers in Artificial Intelligence Research, 2(3), 501-520.

Sun, T., Wang, M., & Chen, J. (2025). Leveraging machine learning for tax fraud detection and risk scoring in corporate filings. Asian Business Research Journal, 10(11), 1-13.

Bian, M., Li, P., Lin, Y., Wang, Y., & Teng, D. (2026). Assessing Green Public Supply Chains Resilience under Carbon Neutrality Goals: A Multi-Agent Reinforcement Learning Framework. IEEE Access.

Teng, D. (2025). TEAS: Token-and energy-aware autoscaling for cost-efficient LLM serving. AI and Data Science Journal, 6(3), 1-10.

Rhee, M., Zou, J., Mo, T., Teng, D., & Yang, J. S. (2026). Enhancing web search agents with self-play contrastive fine-tuning. IEEE Access.

Wang, Z. & Shen. Z. (2024). Flex-Talent: An Explainable And Fair Machine Learning Framework For High-Stakes Leadership-Pipeline Evaluation. World Journal of Management Science, 2(1): 73-82..