Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Mining AI-Assisted Course Design Workflows at Production Scale

Aleksandr Volkov
8Hats Lab https://orcid.org/0009-0000-8834-1843
Taras Pustovoy
CourseFactory and 8Hats Lab https://orcid.org/0009-0003-8298-5113
Dmitriy Istomin
8Hats Lab https://orcid.org/0009-0007-3655-6579
Viacheslav Istomin
8Hats Lab https://orcid.org/0009-0009-3764-134X
Tatiana Otbetkina
t.otbetkina@8hats.ai

8Hats Lab https://orcid.org/0000-0002-9793-2773

Research Article

Keywords: data mining, workfow mining, process mining, AI-assisted course design, learning analytics, production systems, imbalanced classification, privacy-preserving

Posted Date: July 13th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10310358/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Mining AI-Assisted Course Design Workflows at Production Scale

Aleksandr Volkov · Taras Pustovoy · Dmitriy Istomin · Viacheslav Istomin · Tatiana Otbetkina

Abstract We mine CourseFactory, a deployed AI-assisted course-design system, as a multi-surface workflow problem. From a production trace spanning 2023–2026 (197,858 AI calls, 14,598 course versions, 676,417 generated items, 6,323 users), we extract four bounded workflow surfaces using a privacy-preserving protocol that excludes raw prompts, generated text, and calendar timestamps. Version-level structural-quality triage reaches AUPRC 0.89 and saves approximately 20% review effort over random queuing. Item-to-assignment routing achieves a 2.74× AUPRC lift (0.28 vs. 0.10 baseline, AUROC 0.81) on unseen projects. Revision loops identify high-iteration projects for operational review. Task-state conformance against the designed flow reaches 66.8% over the multi-state tail. Feedback and rejected-request surfaces remain diagnostic; withheld input-prompt text adds no measurable signal under lexical, n-gram, and latent-semantic ablations. We formalize the evaluation protocol (Algorithm 1) and release the de-identified dataset, making this the first production-scale workflow-mining study of AI-assisted course design.

Aleksandr Volkov
8Hats Lab
E-mail: av@8hats.ai

Taras Pustovoy
CourseFactory and 8Hats Lab
E-mail: tp@8hats.ai

Dmitriy Istomin
8Hats Lab
E-mail: di@8hats.ai

Viacheslav Istomin
8Hats Lab
E-mail: si@8hats.ai

Tatiana Otbetkina
8Hats Lab
E-mail: t.otbetkina@8hats.ai

Keywords data mining · workflow mining · process mining · AI-assisted course design · learning analytics · production systems · imbalanced classification · privacy-preserving

1 Introduction

AI-assisted course-design systems now generate outlines, assignments, outcomes, and quizzes, yet evidence about their deployed workflows still comes mostly from benchmarks, prototypes, or classroom studies. Production logs expose a different object: the workflow by which users request designs, receive generated artifacts, revise them, hit failures, and give feedback.

We study CourseFactory, a deployed AI-assisted course-design system. Its production dump (2023–2026) includes generated course versions, asynchronous AI requests, token-usage metadata, structural quality checks, user feedback, and process traces. We mine a privacy-preserving representation that excludes raw prompt text, keeping only metadata and aggregates.

Our claim is applied, not architectural: such systems can be mined as workflows, whose value lies not only in generated text but in triage surfaces over usage, structural validity, iteration, feedback, and delay.

We use *workflow* in the event-log sense. A case can be a project, a generated course version, a course-structure item, or a production task; events include AI requests, version creation, structural-quality metrics, feedback, assignment creation, and task-state transitions. We preserve event order through release-local ordinal ranks rather than calendar timestamps. The paper then mines four workflow surfaces: generated-version review, generated-item elaboration, revision loops, and task-state conformance.

We ask three operational questions. **Q1.** What privacy-preserving representation allows production course-design workflows to be mined without exporting prompts, generated text, or direct identifiers? **Q2.** Which workflow surfaces support robust routing or triage under chronological and new-project evaluation, from generated-version review to generated-item assignment routing? **Q3.** Which revision-loop and task-process traces identify human-review or conformance surfaces, and which timing or coverage boundaries prevent stronger operational claims?

We make four contributions:

1. We define and release a privacy-preserving workflow-mining protocol for CourseFactory logs: typed cases, ordered events, de-identified analytic tables, timestamp audit, new-project split, leakage diagnostics, and controlled artifact boundaries without exporting raw prompts, model outputs, direct identifiers, or calendar dates. The public release contains 14,598 course versions across 9,028 projects, 197,858 external AI usage events, and 676,417 course-structure items across 7,929 projects, plus a datasheet, model card, task-safety taxonomy, event-log format, and surface-evaluation rules.

2. We mine the generated-version review surface: metadata-driven structural-quality triage survives chronological and new-project splits, and a counterfactual review-queue replay cuts reviewer effort by about 20% over random or first-in-first-out triage; feedback and rejected-request targets remain leakage-control surfaces.
3. We mine generated-item assignment routing: leakage-free item-to-assignment progression reaches AUPRC 0.28 versus a 0.10 baseline (a 2.74× lift, AUROC 0.81), generalizes to unseen projects, and leaves head-room for community improvement.
4. We mine operational lifecycle surfaces beyond the scored labels: high-iteration projects become a review trigger, observed task-state transitions support a bounded conformance check against the designed production flow (66.8% conform over the multi-state task tail), and the deployment boundaries translate the results into triage, monitoring, audit, and reuse guidance.

The study is a framework-and-dataset contribution rather than a new model architecture: we formalize the workflow-surface abstraction and evaluation protocol, release a production-scale dataset, and demonstrate their joint use on a deployed system. The classifiers are intentionally standard; the contribution is the reusable evaluation discipline that governs them.

2 Related Work

2.1 Educational Data Mining and Learning Analytics

Educational data mining (EDM) and learning analytics (LA) study learner traces to improve educational outcomes Romero and Ventura (2007); Siemens and de Baker (2012). The dominant object is *student* behavior: clickstreams, quiz attempts, forum posts, and time-on-task logs extracted from learning management systems Lockyer and Dawson (2012); Slade and Prinsloo (2013). Review-queue and triage applications in this tradition target learner performance prediction and early-warning systems Fei and Yeung (2015); Tu et al. (2018). Learning analytics dashboards increasingly face *instructors* rather than students, presenting behavioral aggregates to support teaching decisions Paulsen and Lindsay (2024); AlZoubi and Baran (2026); yet these dashboards surface student-activity signals, not the instructor's own production workflow. Semler et al. Semler et al. (2025) survey 3,028 educational process-mining studies and explicitly identify instructor-facing production analytics as an open gap. CourseFactory occupies exactly this gap: its cases are course versions, structure items, and production tasks generated by instructors and designers, not learner interactions.

2.2 Process Mining in Production Systems

Process mining reconstructs, checks, and improves workflows from event logs van der Aalst (2016); Dumas et al. (2018). Van der Aalst's Process Mining Manifesto establishes event logs, discovery, conformance checking, and enhancement as the four core operations van der Aalst et al. (2012). Conformance checking methods range from token-based replay Rozinat and van der Aalst (2008) through cost-based alignment Adriansyah et al. (2011) to template-based micro-process support Tenschert and Lenz (2016); the PM4Py library Berti et al. (2023) and the Inductive Miner family Leemans et al. (2014) provide widely-used implementations. Educational process mining adapts discovery and conformance to learner navigation and assessment sequences Bogarin et al. (2018); Cerezo et al. (2024). Production-system process mining extends the same operations to business workflows with strict privacy and leakage constraints Augusto et al. (2017); privacy-preserving variants have emerged for healthcare Pika et al. (2020) and general enterprise logs Mannhardt et al. (2019); Rafiei et al. (2020), but not yet for AI-assisted authoring systems.

Our task-state conformance check applies the same design-model comparison used in production workflow audits: each observed transition is matched against the transitions encoded in the designed flow (`artefactflowstepnext`), and the coverage gap between observed (50) and designed (1,112) transitions is reported as a readiness bound rather than a model defect. We use simple transition matching rather than alignment-based conformance because 89% of tasks are single-state (no transitions to align) and only 50 of 1,112 designed transitions are observed, making the alignment fitness score uninformative on this snapshot; richer lifecycle data would justify full alignment analysis. This framing follows calls for honest scope declaration in process-mining contributions van der Aalst et al. (2012).

2.3 Workflow Mining and Temporal Evaluation

Temporal correctness is the central validity threat in deployed workflow mining. Chronological splits rather than random splits are standard practice when features and labels co-evolve over a deployment period AlSumait et al. (2008); Bach and Maloof (2008); Masud et al. (2010). Leakage audits that verify feature timestamps precede label timestamps are now recognized as a prerequisite for credible production ML claims Kapoor and Narayanan (2023). Distribution shift between training and deployment periods motivates the new-project evaluation protocol we use alongside chronological splits Huang et al. (2017): a model that generalizes to unseen projects demonstrates it has not memorized project-specific patterns. Sequential and event-log prediction work in data mining Xiao and Eckert (2013); Cuppers and Vreeken (2020) motivates treating cases as ordered sequences rather than independent rows, which is the representation we adopt for version, item, and task surfaces alike.

2.4 Imbalanced Classification and AUPRC Reporting

All three scored surfaces in this paper exhibit label imbalance: the structural quality target has a positive rate of 0.816, assignment routing has a positive rate of 0.118, and feedback is saturated at 0.922. Standard accuracy or macro-averaged metrics misrepresent classifier value under these conditions. Area under the precision–recall curve (AUPRC) is the appropriate primary metric for imbalanced production classification tasks, as it directly measures ranking quality in the minority class and is sensitive to changes at operational thresholds Cieslak and Chawla (2008); Zhang et al. (2021). We complement AUPRC with precision-at-budget operating points (top-1%, top-10%) to express reviewer-effort reduction in deployment terms. Bootstrap confidence intervals over held-out rows, with resampling at the project level for iteration-tail claims, quantify estimate stability Bach and Maloof (2008). Stronger imbalanced learners (SMOTE oversampling, HistGBM) are evaluated as ablations and yield no lift, confirming that the signal is a property of the features rather than a modeling artifact.

2.5 AI-Assisted Authoring and Production ML

AI writing assistance tools have been studied primarily in coding contexts. Longitudinal studies of GitHub Copilot find modest or negligible effects on objective productivity measures despite high perceived utility Stray et al. (2026); Vella and Blincoe (2026), and large-scale log analyses of developer behavior reveal a quiet restructuring of workflows rather than a simple speedup Sergeyuk et al. (2026). In educational authoring, evidence is almost exclusively prototype- or survey-based: AI-in-education reviews document adoption, responsible use, and practical limitations of large language models Kasneci et al. (2023); Yan et al. (2024), but do not mine production workflows at scale. Our study is the first to mine a deployed AI-assisted *course-design* system as a workflow-mining object, complementing the coding-tool literature with a creative authoring domain where task structure, feedback loops, and quality criteria differ substantially from code compilation.

Production ML disciplines reinforce our methodological stance. Breck et al. Breck et al. (2017) and Polyzotis et al. Polyzotis et al. (2017) argue that production ML systems require evidence beyond held-out accuracy: monitoring, leakage audits, and bounded operational claims are the missing layer. Sambasivan et al. Sambasivan et al. (2021) show that undervalued production data creates compounding data cascades that invalidate downstream model claims. Our privacy ladder, timestamp audit, new-project split, leakage diagnostic, and surface-evaluation rules instantiate these principles for a single deployed system, making the protocol reusable for other AI-assisted authoring platforms.

3 System and Data

CourseFactory is a production system for AI-assisted course design. Users create projects, specify pedagogical parameters, and receive generated versions containing course structure and course-design artifacts. It records AI requests, usage metadata, generated versions, structural checks, feedback, assignments, quizzes, task histories, and delay metrics.

3.1 Workflow Representation

We represent CourseFactory as a multi-surface workflow rather than a single flat scoring table (Fig. 1). Project cases generate course-version events; versions contain course-structure item cases; items can later progress to instructor-facing assignment artifacts; repeated versions form revision loops; and production task histories form a separate process surface. The release preserves the order needed to reconstruct these surfaces while removing raw text, direct identifiers, and calendar time.

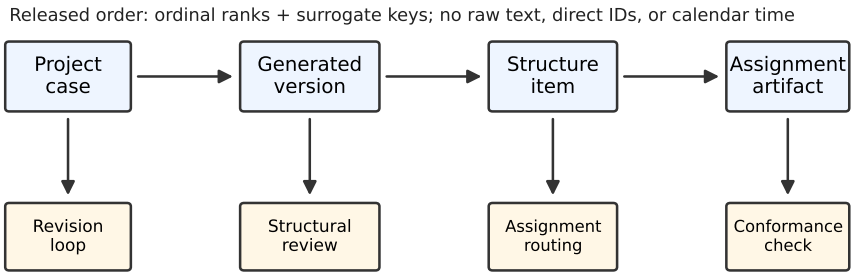


Fig. 1 CourseFactory workflow surfaces mined in the paper. The study follows generated versions into structural review, generated items into assignment artifacts, repeated versions into revision-loop triggers, and task histories into bounded conformance checks.

3.2 Database Snapshot

The snapshot is a PostgreSQL custom dump (2026-04-28) from production; the local restore holds 2,580 table-of-contents entries, and all analysis runs from this restore.

The primary tables are external AI usage (197,858 rows), asynchronous AI requests (197,389), assignments (79,967), assignment quizzes (27,228), the audit log (36,068), project-task state history (22,746), and the outline-builder tables: 9,316 projects, 14,598 versions, 10,295 quality-metric rows, and 3,126

feedback rows. This is the Q1 evidence boundary: we mine version-level workflow metadata while leaving raw prompts, generated content, and broader lifecycle reconstruction outside the released table.

The wider database holds larger process-mining tables: 3,490,858 project-task-assignee delay rows, 3,484,017 artifact delay rows, 2,605,127 document-file-section rows, and 1,690,139 learning-outcome link rows.

Temporal coverage. The main AI usage period spans 2023-05-05 to 2026-04-27. Outline-builder versions span 2023-05-05 to 2026-03-04. Audit events span 2022-11-21 to 2026-04-27. Assignment artifact creation spans 2023-06-15 to 2026-03-04.

3.3 Version-Level Analytic Table

We build a version-level view, one row per generated course version, joining version and project metadata, language, token usage, structural quality checks, and AI-request and feedback aggregates.

It contains 14,598 rows (10,291 with structural quality metrics, 3,126 with feedback, all with AI usage metadata) across 9,028 projects and 9 languages.

Raw prompt text and model output are excluded; retained usage features are prompt length, system-prompt length, token counts, error flags, and result-presence flags.

4 Workflow Surface Labels and Leakage Controls

4.1 Scored Surface Labels and Diagnostics

At the scored-surface layer, labels instantiate the workflow lifecycle: structural quality for generated-version review, assignment progression for generated-item elaboration, and feedback/request state as version-level monitoring or leakage diagnostics. Revision-loop and task-state surfaces are audited below.

Structural quality. The structural quality score is the sum of seven boolean checks recorded for generated course versions: module count validity, learning outcomes in syllabus, item-to-outcome coverage, outcome-to-task coverage, lesson-to-task coverage, no module starting with a task, and study-load consistency. The score ranges from 0 to 7. We model `quality_ge_5`: passing at least five checks is a conservative validity threshold (a majority satisfied, two left for human review). It is not a pedagogical-quality label and assumes no equal weighting of checks.

Assignment progression (item level). For each course-structure item, the routing label records whether it later receives an instructor assignment. Unlike the version-level labels, this label is leakage-free by construction: every assignment is created strictly after its item (0 of 79,967 assignments precede their item), so it needs no feature timestamp audit. Features are item-creation

attributes only: type, position, depth, title and description lengths, language, and a strict-cutoff count of prior items in the version. The snapshot has 676,417 active items across 7,929 projects at an 11.8% assignment rate.

User feedback. Feedback is recorded as positive or negative on generated outline-builder versions. The current snapshot contains 2,883 positive and 243 negative feedback rows.

AI request state. Asynchronous AI requests are recorded as done, rejected, in-progress, or new. The current snapshot contains 194,384 done requests and 2,988 rejected requests at the request-state level. These are distinct from the 321 metric-linked rejected requests used as the version-level diagnostic label; the two populations differ by an order of magnitude and should not be conflated.

4.2 Feature Families

The analytic features are limited to non-text or aggregate fields: language, course duration, study load, knowledge level, practical level, project state, project type, token counts, prompt length, system prompt length, AI usage error flag, AI usage result flag, and request metadata. Raw prompts, generated text, and user-identifying fields are excluded.

The quality label and the feature set are computed from different objects: the structural-quality score is derived from the generated syllabus content, whereas the features are input parameters and aggregate-usage metadata. The only partial overlap is study load, which appears as an input feature and also enters one of the seven structural checks; its permutation importance is low (reported below). This separation guards against circularity: were the quality target a re-encoding of the features, a classifier would approach perfect separation, whereas the observed AUROC near 0.67 is far from that and is consistent with a genuine but modest signal.

4.3 Leakage Protocol

Production databases often encode outcomes in adjacent tables, so we treat leakage control as a first-class method, not a cleanup step.

The reported controls are a temporal train/test split by course-version creation time; a new-project held-out slice that removes training-period projects; feature-family ablations separating course metadata from token, prompt-length, AI-usage, and request-state traces; an explicit feature timestamp audit requiring predictors to be available no later than the scoring point; exclusion of raw model results and quality-check-derived fields; separation of pre-generation forecasting checks from post-generation triage; and explicit flagging of labels best read as leakage diagnostics rather than validated forecasts.

Our claims are scoped accordingly: structural quality as post-generation triage, feedback as descriptive under the current non-text features, and

request rejection as a leakage diagnostic unless stricter lifecycle reconstruction establishes pre-outcome availability.

The timestamp audit supports the post-generation framing of the version-level labels. Quality metrics are always recorded after the course version timestamp in this snapshot: across 10,295 metric rows, zero rows precede the version creation time, the median delay is 2.91 minutes, and the 95th percentile delay is 19.39 minutes. Feedback is also strictly post-version: across 3,126 feedback rows, zero rows precede the version timestamp, the median delay is 7.25 minutes, and the 95th percentile delay is 2,953.25 minutes. These facts make the quality and feedback labels suitable for post-generation triage, but not as predictors for pre-generation course planning.

5 Workflow Surface Evaluation

We define a *workflow surface* as a tuple $\mathcal{S} = \langle C, E, X, y, \rho \rangle$ where C is a case type (e.g. course version, course-structure item, production task), E is the set of events associated with each case, X is the feature family restricted to a privacy tier, y is a candidate label, and $\rho \in \{\text{SCORED}, \text{DIAGNOSTIC}, \text{MONITORING}\}$ is the surface role assigned by the evaluation protocol below (surfaces whose labels lack predictive signal under the current features receive the MONITORING role and serve descriptive or coverage-tracking purposes). A single production system exposes multiple surfaces; the protocol evaluates each independently and assigns bounded operational roles rather than a single aggregate accuracy.

We evaluate each workflow surface with a bounded claim: corpus characterization for release scope, auditable baselines for scored or diagnostic surface labels, temporal/new-project splits for generalization, and process-readiness checks where lifecycle support is too thin for forecasting. Algorithm 1 formalizes this protocol as a reusable procedure applicable to any production AI-assisted authoring system that exposes case-event logs.

Figure 2 illustrates how the five phases of Algorithm 1 connect the raw production dump to the bounded surface assignments reported in Table 2.

5.1 Auditable Surface Baselines

For each scored or diagnostic surface label we compare auditable baselines: majority-class and marginal-frequency, regularized logistic regression, and class-weighted random forests, all under time-split evaluation.

We report balanced accuracy, AUPRC, and AUROC on the new-project chronological slice. With minority outcomes (negative feedback, rejected requests) we emphasize AUPRC over accuracy, since precision-recall is more informative under imbalance Davis and Goadrich (2006).

Categoricals are most-frequent-imputed and one-hot encoded; numerics are median-imputed and standardized. Logistic regression uses balanced weights,

Algorithm 1 Bounded Workflow-Surface Evaluation Protocol

Require: Event log $\mathcal{L}$ with cases C, events E, features X, candidate labels Y
Require: Designed process model $\mathcal{M}$ (optional)
Ensure: Surface role assignment $\mathcal{R} : S \to \{\textsc{scored}, \textsc{diagnostic}, \textsc{monitoring}\}$
Phase 1: Privacy Ladder
for each feature $x_i \in X$ **do**
  Classify x_i into tier: METADATA, AGGREGATE, TEXT, IDENTIFIER
  if $x_i \in \{\textsc{text}, \textsc{identifier}\}$ **then**
    Exclude x_i from released table; retain integer length only
  end if
end for
Replace calendar timestamps with ordinal ranks; replace identifiers with salted surrogates
Phase 2: Leakage Audit
for each candidate label $y_j \in Y$ **do**
  Compute timestamp gap Δt_{ij} between each feature x_i and y_j
  if $\exists x_i : \Delta t_{ij} \leq 0$ **then**
    Flag y_j as DIAGNOSTIC (feature may post-date outcome)
  end if
end for
Phase 3: Temporal and New-Project Splits
Split $\mathcal{L}$ chronologically at quantile q (default 0.80)
Remove training-period cases from test set $\to$ new-project slice
Fit baselines on training set; evaluate on both slices
Phase 4: Bounded Surface Assignment
for each surface s_k with label y_j and evaluation metrics m_k **do**
  if y_j is DIAGNOSTIC **then**
    $\mathcal{R}(s_k) \leftarrow$ DIAGNOSTIC
  else if m_k.AUPRC $>$ positive-rate baseline on new-project slice **and** bootstrap 95% CI excludes baseline **then**
    $\mathcal{R}(s_k) \leftarrow$ SCORED
  else
    $\mathcal{R}(s_k) \leftarrow$ MONITORING
  end if
end for
Phase 5: Process Conformance (if $\mathcal{M}$ provided)
for each observed transition $(a_i \to a_j)$ in multi-state cases **do**
  if $(a_i \to a_j) \in \mathcal{M}$ **then**
    Mark conforming
  else
    Mark deviating; record transition pair and frequency
  end if
end for
Report conformance rate over multi-state tail only
return $\mathcal{R}$, evaluation tables, conformance report

the `liblinear` solver, and 2,000 iterations; random forests use 300 trees, minimum leaf size 10, balanced-subsample weights, and seed 42. We run one training/evaluation pass per label, feature set, and split. To avoid test-set model selection, a time-respecting inner-validation grid (tree count, depth, leaf size) is reported alongside the fixed configuration on the held-out new-project slice; test-slice uncertainty comes from bootstrapping the held-out rows after a single fit.

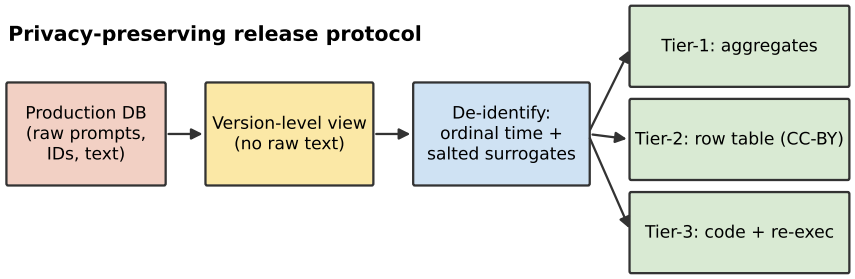


Fig. 2 Workflow-surface evaluation protocol: privacy ladder, leakage audit, temporal/new-project splits, bounded surface assignment, and optional process conformance.

5.2 Temporal and New-Project Evaluation

Baselines use internal version creation order; the release reproduces the split with ordinal ranks rather than calendar timestamps. We report quarterly coverage and label rates first, since the data is not iid: volume peaks in 2024, feedback coverage shifts, and late-2025/2026 quarters are thinner than 2023–2024.

A stricter "new-project" slice removes training-period projects from the test set, checking whether the signal is just repeated versions of one project. For leakage diagnostics we compare three feature sets: full non-text usage, the same without AI usage flags, and course metadata only (no token counts, prompt-length aggregates, usage flags, or current project state).

For structural quality we compute permutation importance on the new-project slice with average precision as the scoring function, keeping the interpretation aligned with the imbalanced metric rather than impurity-based importance, using 20 repeats and seed 42.

Process surface. We treat the delay and task-history tables as a process surface, not a ready-made target: a readiness audit reporting task-type coverage, transition and rollback counts, delay-table coverage, and deadline aggregates, separating feasible lifecycle reconstruction from premature missed-deadline or bottleneck forecasting.

The generated-version models use the version-level analytic table; the generated-item assignment model uses the item-level de-identified table linked back to versions by release-local surrogate keys. Both tables exclude raw prompts, generated text, direct identifiers, and calendar timestamps; text-bearing fields enter only as aggregate metadata or integer lengths.

6 Results

6.1 Corpus and Descriptive Statistics

The corpus is production-scale, not a pilot: 14,598 generated course versions across 9,028 projects and nine languages, with 197,858 external AI usage rows and 197,389 asynchronous AI requests. Request states are dominated by successful completion (194,384 done, 2,988 rejected, 14 in progress, 3 new), usage rows average 932.72 total tokens, and external AI usage spans 2023-05-05 to 2026-04-27. Generated versions are predominantly English (9,693), followed by Russian (1,385), Spanish (1,292), British English (747), Arabic (459), German (413), French (335), Korean (269), and Norwegian (5).

Quality and feedback. The structural-quality score spans the full range and concentrates at the extremes (4,313 versions score 0, 3,349 score 7); feedback is positive-skewed but usable (Table 1).

Table 1 Version-level review and diagnostic label balance.

Target	Eligible	Positive	Share
Quality score ≥ 5	10,291	8,396	0.816
Positive feedback	3,126	2,883	0.922
Rejected metric request	10,291	321	0.031

Structural quality has enough support for review triage; feedback (92.2% positive) serves coverage monitoring only, as chronological baselines below show no detectable signal under non-text features.

Temporal drift. The production trace is temporally nonuniform. Quarterly version volume peaks in 2024Q1 with 2,943 versions and 2024Q2 with 2,754 versions, then falls to 833 versions in 2025Q1 and 198 versions in 2026Q1. Mean structural quality among versions with metrics remains in a relatively narrow band across quarters (5.42–5.93), while positive feedback share is consistently high (0.877–0.982). Figure 3 shows this volume drift against comparatively stable quality/feedback aggregates. This makes chronological evaluation a requirement, not an optional check: a random split would mix deployment periods of different volume and coverage.

Table 2 is the results map: generated-version review appears in two views, structural triage and queue replay; then the lifecycle moves to generated-item elaboration, revision loops, and task-state conformance.

6.2 Generated-Version Surface: Structural-Quality Triage

We run an 80/20 chronological split on non-text features, then repeat after removing training-period projects from the test set (20 overlapping projects for quality and request, 5 for feedback). Across rate baselines, balanced logistic

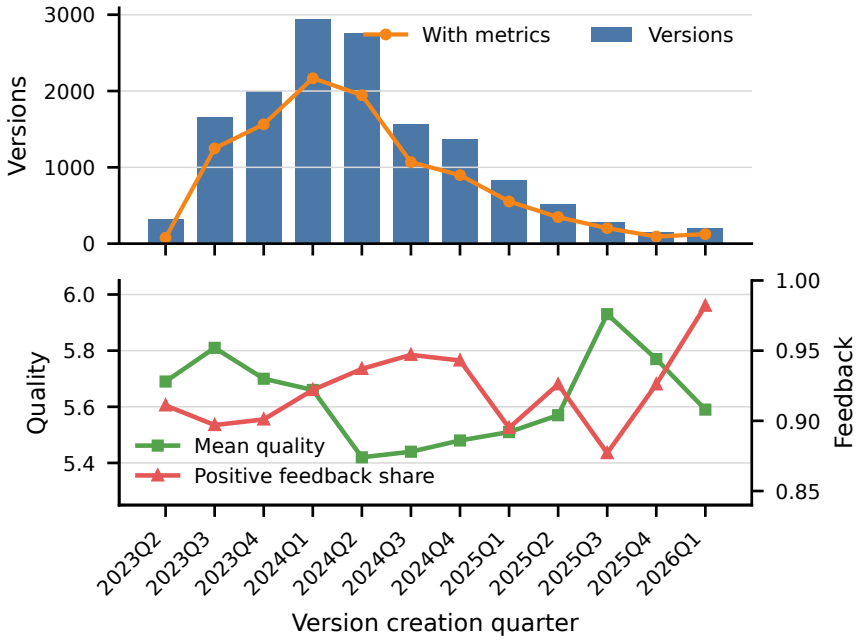


Fig. 3 Quarterly version volume, metric coverage, mean structural quality, and positive feedback share: substantial volume drift with comparatively stable quality and feedback rates.

Table 2 Workflow surfaces mined from CourseFactory. Each surface is a bounded view of the same production course-design workflow.

Surface	Unit	Scale	Use	Result	Boundary
Version review	Version	14,598 ver.	Review queue	AUPRC .890 vs .819	Triage; not grading
Item elaboration	Item	676,417 items	Assign. routing	AUPRC .277 vs .101	Post-dates features
Revision loop	Proj. seq.	78 proj., 11+ ver.	Repeat trigger	Mean quality 4.46	Observational
Task process	Event log	22,746 events	Conformance	66.8% conform.	Multi-state tail

regression, and class-weighted random forests over three feature sets, Table 3 reports the new-project random-forest rows; logistic regression follows the same pattern at slightly lower scores.

Structural quality shows a modest but real ranking signal: full usage and course metadata are close and both clear the rate baseline, indicating the signal is tied to course configuration rather than AI usage failure flags. Feedback shows no detectable signal (underpowered: only 5 new-project test projects). Rejected requests are highly predictable from full usage but collapse under metadata only, confirming their diagnostic status.

Table 3 New-project chronological baseline results. Rejected-request rows are diagnostic leakage checks, not validated forecasting claims.

Label	Model	Bal. acc.	AUPRC	AUROC
Quality ≥ 5	Rate baseline	0.500	0.819	0.500
Quality ≥ 5	RF full usage	0.612	0.890	0.673
Quality ≥ 5	RF metadata	0.617	0.886	0.659
Feedback positive	Rate baseline	0.500	0.924	0.500
Feedback positive	RF metadata	0.538	0.920	0.484
Rejected req. diag.	Rate baseline	0.500	0.024	0.500
Rejected req. diag.	RF full usage	0.987	0.921	0.994
Rejected req. diag.	RF metadata	0.763	0.217	0.811

Table 3 is the generated-version answer to Q2: only the structural-quality surface supports a bounded triage claim under the new-project chronological slice. Full non-text usage and course-metadata-only features track closely across the operating range and both clear the baseline, matching the statistically indistinguishable feature-delta test. The feedback and rejected-request rows are kept in the table because their weaker status is part of the leakage-control result. A bootstrap over the held-out rows with fixed fitted models preserves this reading: the quality-RF intervals stay well above the rate baseline (full-usage AUPRC [0.872, 0.907], AUROC [0.642, 0.705]), feedback-RF remains below its positive-rate baseline, and the rejected-request effect persists only under full usage, reinforcing the diagnostic boundary.

A paired feature-delta bootstrap finds that full usage exceeds metadata-only by only 0.004 AUPRC [−0.010, 0.018] and 0.014 AUROC [−0.015, 0.046]: metadata-only triage is close to full usage, supporting the privacy-preserving release. All bootstrap intervals use 1,000 resamples and the percentile method; iteration-tail intervals below resample projects to respect clustering. Stronger imbalanced learners (SMOTE oversampling, gradient boosting) yield no lift over the class-weighted forest (ΔAUPRC −0.003 and −0.015 respectively), confirming that the signal is a property of the features, not of model choice.

A Platt calibrator reduces the raw forest's Brier score from 0.184 to 0.139 without changing AUROC/AUPRC. For a calibrated metadata-only review queue, the highest-risk 1% of held-out versions contains 40.0% structurally weak cases (2.21× the base rate); the top 10% captures 23.6% of weak cases at 42.6% precision (Table 4). We treat the model as a ranking signal, not a probability estimate, unless a deployment calibration step is applied.

The privacy ladder confirms this empirically: the course-metadata-only tier (9 features) reaches AUPRC/AUROC 0.886/0.659, the non-text aggregate tier 0.889/0.671, and the full non-text tier 0.890/0.673. Almost all structural-triage signal is already present in de-identified course metadata.

We further quantify the cost of text exclusion with an internal-only ablation (no text leaves the analysis environment). Lexical and n-gram features (256 hashed dimensions) from the withheld input prompts of 14,427 versions yield no detectable gain on the structural-quality target (ΔAUPRC −0.004 [−0.018,

Table 4 Precision-at-k operating points for structural-quality triage (metadata-only, calibrated, new-project slice) and item-routing (new-project slice). Base rates: 18.3% weak versions, 11.8% assigned items.

Surface	Budget k	Precision	Recall	Lift
Version quality	Top 1%	0.400	—	2.21×
Version quality	Top 10%	0.426	0.236	2.33×
Item routing	Top 1%	0.450	—	4.45×
Item routing	Top 10%	0.296	0.293	2.51×

0.011] chronological; −0.006 [−0.021, 0.008] new-project) or on feedback (+0.012 [−0.011, 0.035]). A latent-semantic representation (TF-IDF with truncated SVD, 128 components) also shows no gain (−0.004 [−0.020, 0.012]). The claim is bounded to non-neural representations; large neural sentence embeddings remain the one untested source of withheld signal.

The hyper-parameter grid does not overturn the reported configuration: an inner-validated depth-limited forest reaches AUPRC/AUROC 0.883/0.664 on the held-out test, below the fixed configuration's 0.890/0.673. A 10-seed stability run gives mean AUPRC 0.891 (SD 0.0015), and rolling-origin tests show the signal strengthening from 0.848/0.627 at a 50% training cutoff to 0.890/0.673 at the 80% cutoff used in Table 3.

Feature importance. Permutation importance confirms that the structural-quality signal is driven by course-design configuration rather than by direct error flags. In the full usage feature set, duration is the largest feature by average-precision drop (0.030), followed by completion tokens (0.010), project state (0.007), language (0.004), and course study load (0.004). In the metadata-only feature set, duration remains dominant (0.042), followed by language (0.017), course study load (0.010), public availability (0.008), practical level (0.005), and knowledge level (0.004). AI usage error/result flags have zero measured importance for the quality target in this diagnostic. Using the 20 permutation repeats as error bars, the leading full-usage features remain positive: duration has an average-precision drop of 0.030 with interval [0.023, 0.037], completion tokens 0.010 [0.004, 0.015], project state 0.007 [0.001, 0.014], language 0.004 [0.001, 0.007], and course study load 0.004 [0.000, 0.008]. Several lower-ranked token and prompt-length features have intervals crossing zero, so we do not interpret them individually.

6.3 Generated-Version Surface: Review-Queue Replay

To close the generated-version surface, we test whether the triage ranking saves reviewer effort by replaying a weakness-review queue over the held-out version stream. The queue surfaces structurally weak versions (score below 5) for human review, ranked by predicted weakness, and we compare it against the two queues a team could deploy without any model: random order and first-in-first-out (chronological). On the new-project slice (weak rate 0.183),

the metadata queue catches 23.9% of weak versions within the first 10% of versions reviewed, against 10.3% for random and 12.0% for chronological order. To catch 80% of weak versions, the model queue reviews 62.4% of versions versus 78.5% for random, a 20.6% reduction in review effort; its normalized triage efficiency is 0.29 on a scale where random is 0 and a perfect oracle is 1, and the chronological slice is nearly identical (19.1% effort saved). This is a counterfactual replay on historical data, not a live deployment, but it bounds the operational value of the signal against the baselines a team would otherwise use, and it runs from the released table alone (Fig. 4).

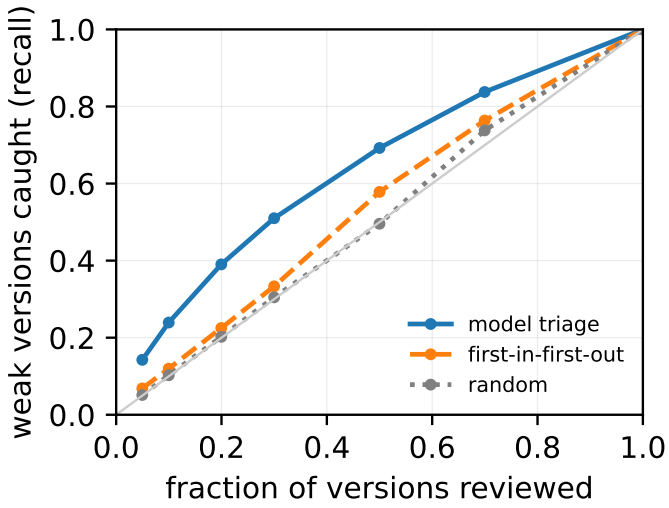


Fig. 4 Review-queue replay (new-project slice): the metadata queue (blue) catches weak versions faster than first-in-first-out (orange) or random (dotted); to reach 80% recall it reviews 62% of versions versus 79% for random, a 20.6% effort reduction.

6.4 Generated-Item Surface: Assignment Routing

After version review, generated-item elaboration follows course-structure items into instructor-facing assignment artifacts. Given a course-structure item, we score its later progression to an instructor assignment. The routing label is leakage-free by construction, because every assignment timestamp post-dates its item (0 of 79,967 assignments precede their item), and it uses only item-creation features (type, position, depth, title and description lengths, language, and a strict-cutoff count of prior items in the version); as elsewhere, only integer text lengths are read, never raw text. The item-level surface spans 676,417 active items across 7,929 projects at an 11.8% positive rate, an order of magnitude more instances than the version-level labels.

A class-weighted random forest reaches AUPRC 0.277, 95% interval [0.270, 0.283], on the new-project held-out slice against a 0.101 positive-rate baseline (a 2.74× lift) with AUROC 0.809; the chronological slice is equivalent (AUPRC 0.281 [0.275, 0.289], AUROC 0.811), so the signal generalizes to unseen projects rather than memorizing them. As a routing queue, the top 1% of items by score has 45.0% precision (4.45× the base rate) and the top 10% has 29.6% precision at 29.3% recall. Item type is the strongest feature, but the result is not a type lookup: within each assignable type the model still separates classes (AUROC 0.68, 0.63, and 0.61 for the three assignable types), and a Platt calibrator reduces Brier score from 0.153 to 0.082. Unlike the saturated structural-quality surface, item routing leaves head-room for community improvement, so it is the flagship routing surface in the released workflow artifact (Fig. 5).

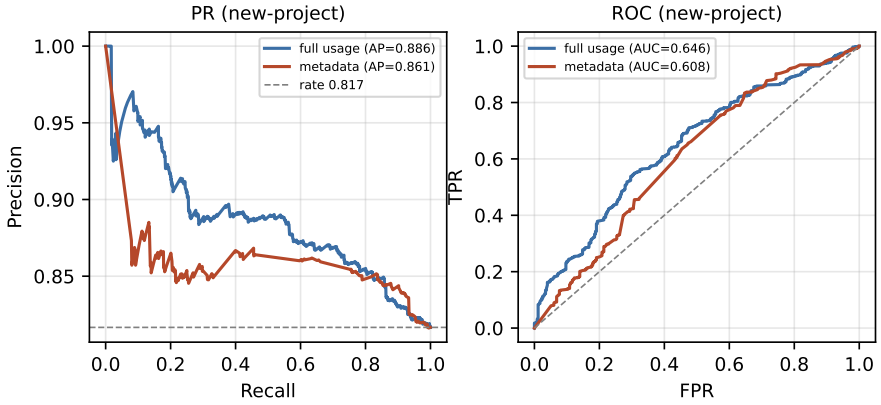


Fig. 5 Precision–recall and ROC curves for the generated-item assignment routing surface on the new-project held-out slice. The PR curve shows AUPRC 0.28 against the 0.10 baseline; the ROC curve shows AUROC 0.81.

Metric-request timing. The rejected-request label remains diagnostic: of 7,523 metric rows linked to AI requests (7,202 done, 321 rejected), rejected requests are created shortly after the version timestamp (median 0.580 min, P95 2.109 min), but 1,045 done request rows precede the version timestamp. These timings do not support treating request-rejection status as pre-outcome forecasting.

6.5 Revision-Loop Surface: Project Iteration

Most projects have a single generated version, but a meaningful tail repeats: 78 projects accumulate at least eleven versions each, 1,457 versions in total (18.68 on average), motivating a separate analysis of high-iteration cases rather than treating every version as independent (Table 5).

Iteration count is also a diagnostic surface. Mean structural quality declines monotonically across version buckets while the rejected-version share peaks in the 6–10 bucket (Table 5); among 11+ projects with metric rows at both endpoints, 21 worsen, 10 improve, and 7 are unchanged. The applied reading is not that repeated generation causes lower quality, but that it marks projects needing workflow triage or human review.

Table 5 Iteration-tail diagnostics by project version count. Mean quality is over versions with metric rows; ΔQ is the mean first-to-last quality change where both endpoints have metric rows.

Bucket	Proj.	Vers.	Mean Q	ΔQ	Fb+	Rej.
1	6,803	6,803	5.85	0.00	0.948	0.002
2	1,273	2,546	5.62	-0.19	0.920	0.046
3	403	1,209	5.48	-0.21	0.876	0.061
4–5	287	1,254	5.35	-0.22	0.892	0.099
6–10	184	1,329	5.25	-0.38	0.785	0.126
11+	78	1,457	4.46	-0.76	0.825	0.039

Table 5 answers Q3 as an operational trigger rather than a causal model. As a concrete operating point, the 11+ threshold sends 78 of 9,028 projects to review, a 0.86% project-level queue that covers 1,457 of 14,598 generated versions. The deployment action is to route this low-volume, high-yield tail for human or template review, while preserving the boundary that retries are an observational marker. Project-resampled intervals reinforce the tail reading: the 11+ bucket has mean quality 4.46 with 95% interval [3.95, 4.99], and its paired first-to-last quality change is -0.76 [-1.29, -0.24]. The 6–10 bucket also has a negative paired interval, -0.38 [-0.72, -0.05], while the 3 and 4–5 buckets cross zero. The quality gap between single-version projects (mean 5.85) and the 11+ tail (mean 4.46) corresponds to an approximate Cohen's $d \approx 0.66$ (using the version-level pooled SD ≈ 2.1; bucket-level SDs are not reported because the quality score is discrete 0–7 and within-bucket variance differs), a medium-to-large effect. The evidence therefore supports review of the high-iteration tail without claiming that retries themselves cause quality loss (Fig. 6).

We also test whether the high-iteration tail can be forecast from first-version metadata. That model is deliberately reported as a negative result: a metadata-only random forest for eventual 11+ project membership reaches average precision 0.0068 and AUROC 0.321 on a project-chronological test, below a training-rate baseline AP of 0.0078. The useful stopping surface is instead a simple runtime count trigger. Once a project reaches 8 generated versions, only 150 projects are queued and all 78 eventual 11+ projects are covered, with 52.0% tail precision. At 10 versions, the queue is 87 projects and still covers all 78 eventual 11+ projects, with 89.7% precision. This is not early semantic forecasting; it is an intervention rule for projects already exhibiting repeated generation behavior.

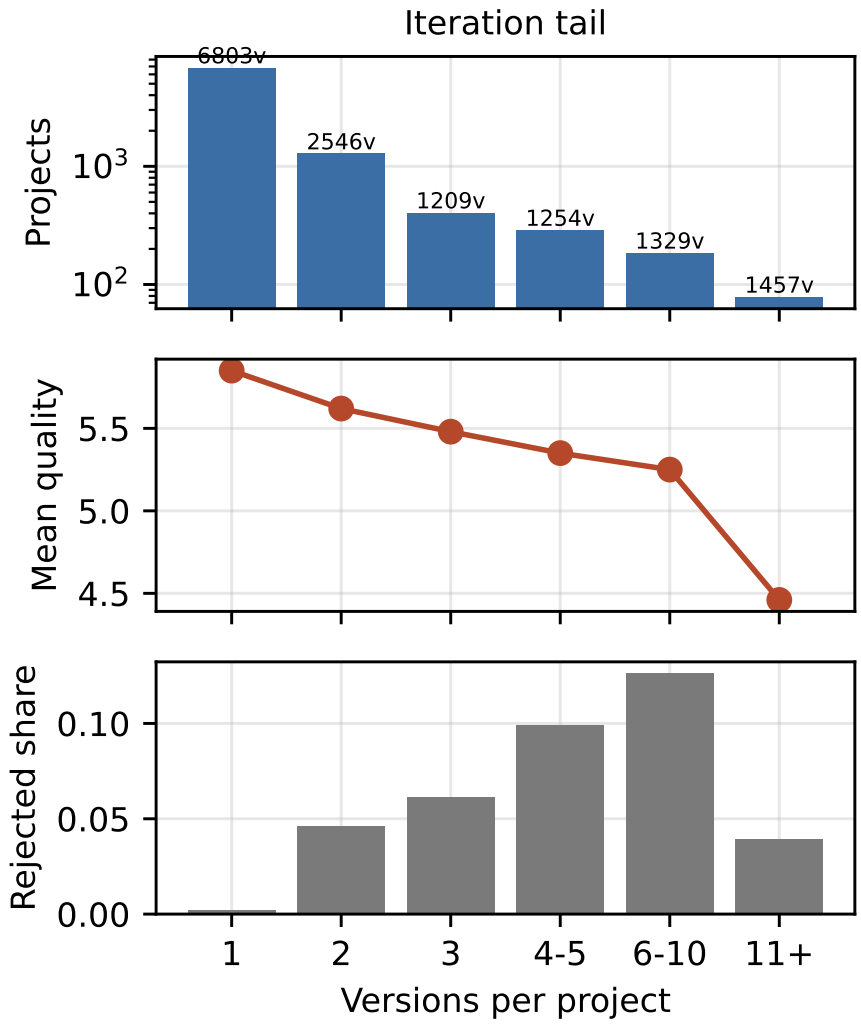


Fig. 6 Mean structural quality and first-to-last quality change by project iteration bucket. Quality declines monotonically with iteration count; the 11+ tail shows the strongest negative delta.

6.6 Task-State Surface: Variant Analysis, Conformance, and Event-Log Release

The production-task lifecycle exposes a process-mining surface distinct from the version-generation surfaces above. The project-task table contains 16,843 tasks across 69 projects. Two task types dominate: artifact-production tasks (8,806; 52.3%) cover content drafting, review, and approval, while video-shooting tasks (6,861; 40.7%) cover media production. The remaining

1,176 tasks (7.0%) belong to secondary types including coordination and administrative steps.

Lifecycle density and the single-state tail.

The task-state history table contains 22,746 recorded state events over all 16,843 tasks. The per-task distribution is heavily right-skewed: median one recorded state, 75th percentile two, 95th percentile three, and a maximum of 38. Of the 16,843 tasks, 14,986 (89.0%) are single-state in the snapshot, meaning their lifecycle was either not yet advanced beyond the initial state or not captured at the time of the dump. The remaining 1,857 multi-state tasks generate 5,903 transition instances and form the analysis unit for conformance and variant work below. The delay tables are dense but entity-concentrated: 3,484,017 artifact-delay rows cover 3 projects and 173 task–assignee links, while 3,490,858 task-assignee delay rows span 604 assignees. This concentration prevents delay-based forecasting on this snapshot; we treat them as descriptive evidence.

Transition timing.

Among multi-state tasks, three transitions dominate by volume and carry interpretable timing (Table 6). The *Ready→Done* transition represents the completion step: 741 instances with a median duration of 0.235 hours, indicating rapid handoff once a task is marked ready. The *New→Ready* transition, which signals that assigned work has been picked up, has a much longer median of 43.787 hours, reflecting the human scheduling lag between assignment and start. The *Reopen→Ready* transition (666 instances, median 1.660 hours) represents rework cycles: tasks returned for revision are picked up substantially faster than new tasks, suggesting rework is treated with higher priority. These three transitions alone account for 37.3% of all 5,903 transition instances.

Table 6 Most frequent task-state transitions with timing in the multi-state tail (1,857 tasks, 5,903 transition instances).

Transition	Count	Median (h)	Share (%)
Ready→Done	741	0.235	12.6
New→Ready	694	43.787	11.8
Reopen→Ready	666	1.660	11.3
Other (47 types)	3,802	—	64.4

Variant distribution.

The 1,857 multi-state tasks produce 227 distinct lifecycle variants. Table 7 reports the cumulative coverage of the top variants. The most common variant

accounts for 61.9% of all tasks in the log; the top three variants together cover 86.1%. The long tail of 224 remaining variants shares the final 14.4%, reflecting idiosyncratic paths that resist a single process template. Variant lengths range from one transition (the dominant paths) to three or more transitions for tasks that cycle through *Reopen* loops. The 226 rollback events (3.8% of all transitions) concentrate in the longer variants and co-occur with the *Reopen*→*Ready* transition, confirming that rework cycles are the primary source of lifecycle complexity.

Table 7 Cumulative coverage of the top task-lifecycle variants (16,843 total tasks; 227 distinct variants).

Top-k variants	Tasks covered	Cumul. share (%)
1	10,424	61.9
2	12,546	74.5
3	14,504	86.1
6	15,072	89.5
10	15,340	91.1
All 227	16,843	100.0

Conformance against the designed flow.

CourseFactory encodes its intended production workflow directly in relational tables: `artefactflowstep` (486 rows) defines the valid states, and `artefactflowstepnext` (1,112 rows) defines the allowed transitions. We check each observed transition in the multi-state tail against this designed model. Of 5,903 observed transition instances, 3,942 (66.8%) match a designed transition; the remaining 1,961 (33.2%) deviate. Only 50 of the 1,112 designed transitions are observed at all, indicating that the snapshot covers a narrow slice of the designed state space.

We report this as a *bounded* conformance result rather than a global process-quality claim. The bound has two sources. First, the 14,986 single-state tasks are structurally excluded: they contribute no transitions and therefore neither conform nor deviate, so the 66.8% rate describes the multi-state tail only. Second, the designed model is permissive at 1,112 transitions, but only 50 are empirically observed; many designed paths may represent rarely-triggered branches, making full conformance over all 1,112 an unreachable ceiling even for a perfectly-operated system.

The top deviating transition pairs account for a disproportionate share of non-conforming instances. The two most frequent deviating transitions together contribute 1,208 instances (61.6% of all deviations), suggesting that a small set of recurring off-script patterns drives most of the non-conformance rather than diffuse random errors. These patterns are interpretable as operational shortcuts: steps executed out of order under deadline pressure or as implicit parallel handoffs not captured in the designed sequential model.

We retain the deviation rows as a monitoring surface for operational process improvement rather than treating them as errors requiring correction.

Event-log release and community reuse.

We export the process surface as a privacy-preserving event log for community reuse. The released log contains all 22,746 state events over 16,843 task cases and 69 project groups. Each row records a surrogate `case_id`, a surrogate `project_case_id`, within-case `event_order`, `hours_since_case_start` (replacing calendar timestamps), categorical `task_type` and `activity_type`, human-readable `activity_name`, and binary `is_rollback` and `is_rollback_possible` flags. Direct task, assignee, organization, and project identifiers are excluded, as are calendar timestamps. This gives process-mining researchers a safe event-log object supporting lifecycle-readiness inspection, variant analysis, and conformance checking against externally defined process models, while keeping full process discovery and bottleneck modelling in scope for future work with denser lifecycle data.

7 Discussion

7.1 Workflow Surfaces Carry Different Epistemic Status

The central finding is not a single number but a differentiation across surfaces. Generated-version structural quality is the strongest version-level review target, with a metadata-driven triage surface that survives chronological and new-project evaluation (AUPRC 0.89, 20% review-effort reduction). Generated-item assignment routing is the strongest item-level routing surface: leakage-free, 2.74× AUPRC lift over base rate, and robust to unseen projects (AUPRC 0.28 vs 0.10 baseline). The revision-loop trigger is operational but observational: high-iteration count is a reliable review signal without implying that retries cause quality loss. The task-state conformance result (66.8%) is bounded to the multi-state tail and supports process-readiness inspection rather than bottleneck forecasting. Feedback and rejected-request surfaces remain descriptive or diagnostic until lifecycle timing is better reconstructed.

This stratification matters because it resists a common failure mode in applied data mining: reporting a single accuracy number and implying it applies uniformly. Each surface has a different label density, leakage profile, and temporal support. The protocol contribution is precisely the discipline of assigning bounded roles to each surface rather than maximizing headline metrics.

7.2 Relation to AI Authoring Tool Studies

Longitudinal studies of AI coding assistants find a paradox: high perceived productivity but modest objective change Stray et al. (2026); Vella and

Blincoe (2026), characterized as a quiet restructuring of workflows rather than a speedup Sergeyuk et al. (2026). Our structural quality signal provides indirect evidence for course design: course duration, language, and study-load configuration—choices made before AI generation—are the dominant predictors, not AI usage flags. This parallels the coding-tool finding that developer skill and task selection moderate AI value more than call volume. The absence of a measurable text-prompt signal further supports the view that metadata-level workflow structure dominates text-level generation behavior for quality triage; whether this holds under large neural sentence embeddings remains the principal open question.

7.3 Process Conformance as a Production Health Metric

The 66.8% conformance rate has two readings: one-third of transitions deviate (pessimistic), but the two most frequent deviating pairs account for 61.6% of all deviations, suggesting a concentrated, correctable gap (optimistic). Operational shortcuts under deadline pressure are well-documented in production workflow mining van der Aalst et al. (2012). For AI-assisted content production, this gap is a monitoring signal rather than a failure indicator: deviations may reflect legitimate expertise or implicit parallelism. Tracking conformance over time—supported by the event-log format once calendar timestamps are available—would distinguish stable shortcuts from drifting compliance.

7.4 Community Workflow-Mining Protocol

The community-facing surfaces of the study turn a private production dump into a governed workflow-mining release: a version-review triage surface, an item-routing surface, a privacy frontier, a responsible-use baseline, an operational stopping trigger, a process-mining event-log, and explicit rules for which surfaces should remain non-scored. Their operating boundaries are summarized in Table 8.

The release is intentionally not maximalist: it defines a few reusable surfaces and assigns non-scored roles to surfaces where timing, privacy, or target semantics are too weak. That restraint is part of the contribution, showing how to mine a production AI-assisted course-design workflow without turning every field into an unsafe forecasting surface. Sambasivan et al. Sambasivan et al. (2021) describe how undervalued production data creates data cascades that invalidate downstream models; our datasheet, model card, and surface-evaluation rules are designed to make such cascades visible before they compound.

7.5 Claim Boundaries

The results should be read through four claim boundaries. First, assignment routing is artifact routing, not content grading; structural-quality and iteration-tail surfaces identify versions or projects for review, not causal effects or automated judgments. Second, the corpus is one deployed system, so the transferable contribution is the privacy-preserving workflow-mining protocol: de-identified tables, timestamp audit, new-project split, leakage diagnostic, and controlled release boundary rather than a universal benchmark result. Third, the privacy boundary trades semantic coverage for release feasibility: raw text exclusion prevents prompt/content-quality analysis, but makes the metadata workflow study auditable and reproducible. Fourth, feedback and rejected-request targets are reported as descriptive or audit surfaces unless future lifecycle reconstruction proves stronger pre-outcome availability.

7.6 Deployment Implications and Future Work

The current evidence supports the staged deployment surface in Table 8: each row pairs a workflow surface with a permitted use and a boundary, converting results into triage, monitoring, and audit actions.

Table 8 Operational use boundaries for CourseFactory workflow surfaces.

Workflow surface and supported use	Boundary
Generated-version review: route versions to structural review.	Not automated grading.
Generated-item elaboration: route items to assignment artifacts.	Post-item label; not content-quality grading.
Revision loop: trigger template, prompt, or configuration review.	Observational; no causal retry claim.
Feedback monitoring: track satisfaction coverage and quality strata.	Descriptive under current non-text features.
Request-state diagnostics: audit instrumentation and failure workflows.	Diagnostic until pre-outcome timing is proven.
Task-state process: reconstruct state and delay lifecycles before bottleneck mining.	Delay rows are concentrated; not modeled here.

Three concrete directions follow from the current boundaries. First, large neural sentence embeddings applied to the withheld input prompts are the most likely source of additional structural-quality signal; our lexical and latent-semantic ablations bound the cost of text exclusion only for non-neural representations. Second, denser lifecycle reconstruction—collecting calendar timestamps and full task-state histories in an ongoing deployment rather than a single snapshot—would enable conformance tracking over time, bottleneck prediction from delay tables currently too concentrated for modeling, and missed-deadline forecasting currently blocked by the absence of

planned-deadline fields in the snapshot. Third, the workflow-mining protocol is designed to be portable: other AI-assisted authoring platforms (presentation generation, document drafting, code review pipelines) expose structurally similar surfaces (generation request, artifact creation, review, approval, rework) that the same privacy ladder, leakage audit, and bounded-claim discipline can govern.

8 Limitations

This study has five limitations that bound the scope of the claims and define directions for future validation.

Single-system scope. The snapshot covers one deployed system (CourseFactory) over one operational period (2023–2026). The transferable contribution is the workflow-mining protocol, not the specific numeric results; whether the same surfaces carry signal in other AI-assisted authoring platforms remains an open empirical question.

Temporal leakage risk. Production schemas often contain fields created after the outcome. Our chronological split and new-project split guard against the most common leakage patterns, and we restrict claims to post-generation triage rather than pre-generation content-quality forecasting. Nevertheless, rejected-request models remain diagnostic because linked usage and request rows are effectively simultaneous: stricter timing reconstruction—requiring per-field creation timestamps not available in the current dump—would be needed to promote these surfaces to validated pre-outcome predictors.

Text exclusion and the neural-embedding gap. The release excludes raw prompts and generated text for privacy. Our ablation measures the cost of this exclusion for three representation families: lexical (bag-of-words), n-gram (character and word), and latent-semantic (truncated SVD over TF-IDF). In all three cases the added signal is indistinguishable from noise, confirming that metadata-level features dominate under these representations. However, large pre-trained sentence encoders (e.g., Sentence-BERT, instructor embeddings) and generative quality estimators (e.g., LLM-as-judge) operate in a fundamentally different feature space that our ablation does not cover. The value of richer text representations behind the privacy boundary therefore remains the principal open question. Answering it would require either a controlled on-premise deployment with consent-based text access, or a synthetic-text proxy validated against the real distribution—both non-trivial extensions.

Sparse and observational surfaces. Feedback is heavily imbalanced (positive-skewed, low volume) and serves descriptive monitoring rather than predictive modeling under the current feature set. The iteration tail is observational: high retry counts correlate with lower mean structural quality, but the causal direction is ambiguous—retries may reflect difficult requirements, changing user intent, prompt-template mismatch, or interface friction. These surfaces support triage and monitoring actions, not standalone outcome forecasting or a causal retry-quality claim.

Conformance baseline without temporal tracking. The 66.8% conformance rate is a cross-sectional snapshot, not a trend. Without calendar timestamps in the released event log, we cannot track whether conformance improves or degrades over time, distinguish stable operational shortcuts from drifting compliance, or correlate conformance with external events (feature releases, team changes, policy updates). A future release with relative monthly bins would enable longitudinal conformance monitoring while preserving the privacy boundary.

9 Ethics, Privacy, and Reproducibility

9.1 Privacy Boundary

The production database holds raw user and system prompts, generated text, and user/organization identifiers, all treated as restricted. The analytic tables exclude these, using text-bearing tables only through aggregate metadata: token counts, prompt lengths, result-presence flags, and request timestamps.

9.2 Reproducibility Artifacts

The analysis is a local, scriptable pipeline over a PostgreSQL restore. SQL produces row-count, schema, temporal, target-balance, and lifecycle-audit outputs; Python scripts run the baselines, ablations, new-project evaluation, permutation importance, calibration, stability, process-readiness, and runtime checks. The artifact records the macOS/PostgreSQL/Python/TeX environment, pipeline runtimes, and evidence files for every numeric claim.

9.3 Data Availability

The raw production dump is not released: it contains private user and organizational data, raw prompts, system prompts, and generated course text. With data-owner approval, we release a public de-identified workflow artifact: aggregate outputs, code and SQL, a row-level version table for all 14,598 generated course versions, and an item-level table for all 676,417

course-structure items.[1] The public tables contain no raw text, direct identifiers, calendar timestamps, or public-availability flag; ordinal time, per-project indices, and salted release-local surrogate keys reproduce the chronological/new-project splits and join item rows back to version rows without a production lookup. The release includes a datasheet, manifest, model card, surface-evaluation rules, event-log format, privacy ladder, task-safety taxonomy, starter script for the structural-quality triage table, and smoke-test synthetic rows that are explicitly not metric evidence.

9.4 Responsible Use

The workflow-mining results are for triage and audit, not automated grading of instructors or learners. The strongest reusable routing surface is item-level assignment progression; structural-quality triage remains a secondary metadata-driven review surface, feedback stays descriptive, and rejected-request status is a leakage diagnostic until stricter request-lifecycle reconstruction proves pre-outcome availability.

10 Conclusion

We present CourseFactory as a production workflow-mining problem for AI-assisted course design. The study makes two contributions: a reusable privacy-preserving workflow-mining protocol and a production-scale dataset release for the research community.

The protocol comprises de-identification with salted surrogate keys, exclusion of raw prompts and generated text, chronological and new-project evaluation splits, a leakage diagnostic for each surface, and bounded surface roles that separate triage and monitoring actions from automated grading. Applied to three years of production data (197,858 AI calls, 14,598 course versions, 676,417 generated items, 22,746 task-state events across 6,323 users), the protocol yields four workflow surfaces with distinct epistemic status:

- *Version-level structural quality* supports bounded review triage with AUPRC 0.89 and approximately 20% review-effort reduction under queue replay.
- *Item-to-assignment progression* is the strongest leakage-free routing surface (AUPRC 0.28 against a 0.10 baseline, AUROC 0.81 on unseen projects).
- *Revision loops* expose high-iteration projects for human or template review; the 10-version trigger covers all 78 projects with 11+ versions.
- *Task-state conformance* reaches 66.8% over the multi-state tail, with two transition pairs accounting for 61.6% of all deviations.

[1] Archived with a persistent DOI; data under CC-BY-4.0, code under a permissive license. DOI: `10.5281/zenodo.20517429`.

Feedback remains descriptive under the current non-text feature set, rejected-request status remains a leakage-audit diagnostic, and withheld input-prompt text adds no measurable lift under lexical, n-gram, and latent-semantic representations. The principal open question is whether large neural sentence embeddings would change this picture behind the privacy boundary.

The released dataset, workflow-mining protocol, model card, and surface-evaluation rules are available at `https://doi.org/10.5281/zenodo.20517429`. We hope this release enables reproducible research on AI-assisted authoring workflows and provides a template for responsible mining of similar production systems.

References

van der Aalst WMP (2016) Process mining: Data science in action. Springer DOI 10.1007/978-3-662-49851-4

van der Aalst WMP, Adriansyah A, de Medeiros AKA, et al. (2012) Process mining manifesto. In: Business Process Management Workshops, Lecture Notes in Business Information Processing, vol 99, Springer, pp 169–194, DOI 10.1007/978-3-642-28108-2_19

Adriansyah A, van Dongen BF, van der Aalst WMP (2011) Conformance checking using cost-based fitness analysis. In: 15th IEEE International Enterprise Distributed Object Computing Conference, IEEE, pp 55–64, DOI 10.1109/EDOC.2011.12

AlSumait L, Barbara D, Domeniconi C (2008) On-line LDA: Adaptive topic models for mining text streams with applications to topic detection and tracking. In: 2008 Eighth IEEE International Conference on Data Mining, IEEE, pp 3–12, DOI 10.1109/ICDM.2008.140

AlZoubi D, Baran E (2026) From emergency response to sustainable practice: Instructional designers' human-centered approaches to online learning. TechTrends 70:702–713, DOI 10.1007/s11528-026-01186-1

Augusto A, Conforti R, Dumas M, La Rosa M (2017) Split miner: Discovering accurate and simple business process models from event logs. In: 2017 IEEE International Conference on Data Mining, IEEE, pp 1–10, DOI 10.1109/ICDM.2017.9

Bach SH, Maloof MA (2008) Paired learners for concept drift. In: 2008 Eighth IEEE International Conference on Data Mining, IEEE, pp 23–32, DOI 10.1109/ICDM.2008.119

Berti A, van Zelst SJ, Schuster D (2023) pm4py: A process mining library for Python. Software Impacts 17:100556, DOI 10.1016/j.simpa.2023.100556

Bogarin A, Cerezo R, Romero C (2018) A survey on educational process mining. WIREs Data Mining and Knowledge Discovery 8(1):e1230, DOI 10.1002/widm.1230

Breck E, Cai S, Nielsen E, Salib M, Sculley D (2017) The ML test score: A rubric for ML production readiness and technical debt reduction. In:

2017 IEEE International Conference on Big Data, IEEE, pp 1123–1132, DOI 10.1109/BigData.2017.8258038

Cerezo R, Bogarín A, Esteban M, Romero C (2024) Process mining for self-regulated learning assessment in e-learning. In: arXiv preprint, 2403.12068

Cieslak DA, Chawla NV (2008) Start globally, optimize locally, predict globally: Improving performance on imbalanced data. In: 2008 Eighth IEEE International Conference on Data Mining, IEEE, pp 143–152, DOI 10.1109/ICDM.2008.87

Cuppers J, Vreeken J (2020) Just wait for it... mining sequential patterns with reliable prediction delays. In: 2020 IEEE International Conference on Data Mining (ICDM), IEEE, pp 82–91, DOI 10.1109/ICDM50108.2020.00017

Davis J, Goadrich M (2006) The relationship between precision-recall and ROC curves. In: Proceedings of the 23rd International Conference on Machine Learning, Association for Computing Machinery, pp 233–240, DOI 10.1145/1143844.1143874

Dumas M, La Rosa M, Mendling J, Reijers HA (2018) Fundamentals of Business Process Management, 2nd edn. Springer, DOI 10.1007/978-3-662-56509-4

Fei M, Yeung DY (2015) Temporal models for predicting student dropout in massive open online courses. In: 2015 IEEE International Conference on Data Mining Workshop, IEEE, pp 256–263, DOI 10.1109/ICDMW.2015.174

Huang B, Zhang K, Zhang J, Sanchez-Romero R, Glymour C, Scholkopf B (2017) Behind distribution shift: Mining driving forces of changes and causal arrows. In: 2017 IEEE International Conference on Data Mining (ICDM), IEEE, pp 913–918, DOI 10.1109/ICDM.2017.114

Kapoor S, Narayanan A (2023) Leakage and the reproducibility crisis in machine-learning-based science. Patterns 4(9):100804, DOI 10.1016/j.patter.2023.100804

Kasneci E, Sessler K, Kuchemann S, Bannert M, Dementieva D, Fischer F, Gasser U, Groh G, Gunnemann S, Hullermeier E, et al. (2023) ChatGPT for good? on opportunities and challenges of large language models for education. Learning and Individual Differences 103:102274, DOI 10.1016/j.lindif.2023.102274

Leemans SJJ, Fahland D, van der Aalst WMP (2014) Discovering block-structured process models from event logs containing infrequent behaviour. In: Business Process Management Workshops, Springer, Lecture Notes in Business Information Processing, vol 171, pp 66–78, DOI 10.1007/978-3-319-06257-0_6

Lockyer L, Dawson S (2012) Where learning analytics meets learning design. In: Proceedings of the 2nd International Conference on Learning Analytics and Knowledge, Association for Computing Machinery, pp 14–15, DOI 10.1145/2330601.2330609

Mannhardt F, Koschmider A, Baber N, Oberweis SA (2019) Privacy-preserving process mining. Business & Information Systems Engineering 61(5):595–614, DOI 10.1007/s12599-019-00613-3

Masud MM, Chen Q, Khan L, Aggarwal C, Gao J, Han J, Thuraisingham BM (2010) Addressing concept-evolution in concept-drifting data streams. In: 2010 IEEE International Conference on Data Mining, IEEE, pp 929–934, DOI 10.1109/ICDM.2010.160

Paulsen L, Lindsay E (2024) Learning analytics dashboards are increasingly becoming about learning and not just analytics—A systematic review. Education and Information Technologies 29(11):14279–14308, DOI 10.1007/s10639-023-12401-4

Pika A, Wynn MT, Budiono S, ter Hofstede AHM, van der Aalst WMP, Reijers HA (2020) Privacy-preserving process mining in healthcare. International Journal of Environmental Research and Public Health 17:1612, DOI 10.3390/ijerph17051612

Polyzotis N, Roy S, Whang SE, Zinkevich M (2017) Data management challenges in production machine learning. In: Proceedings of the 2017 ACM International Conference on Management of Data, Association for Computing Machinery, pp 1723–1726, DOI 10.1145/3035918.3054782

Rafiei M, von Waldthausen L, van der Aalst WMP (2020) Supporting confidentiality in process mining using abstraction and encryption, available at `https://www.researchgate.net/publication/338432872`

Romero C, Ventura S (2007) Educational data mining: A survey from 1995 to 2005. Expert Systems with Applications 33(1):135–146, DOI 10.1016/j.eswa.2006.04.005

Rozinat A, van der Aalst WMP (2008) Conformance checking of processes based on monitoring real behavior. Information Systems 33:64–95, DOI 10.1016/j.is.2007.07.001

Sambasivan N, Kapania S, Highfill H, Akrong D, Paritosh P, Aroyo LM (2021) "everyone wants to do the model work, not the data work": Data cascades in high-stakes AI. In: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, Association for Computing Machinery, pp 1–15, DOI 10.1145/3411764.3445518

Semler RF, et al. (2025) Educational process mining: literature classification, gaps, and emerging opportunities. Frontiers in Education 10, DOI 10.3389/feduc.2025.1543761

Sergeyuk A, Huang E, Karaeva D, Serova A, Golubev Y, Ahmed I (2026) Evolving with AI: A longitudinal analysis of developer logs. In: Proceedings of the 48th IEEE/ACM International Conference on Software Engineering (ICSE), `2601.10258`

Siemens G, de Baker RSJ (2012) Learning analytics and educational data mining: towards communication and collaboration. In: Proceedings of the 2nd International Conference on Learning Analytics and Knowledge, Association for Computing Machinery, pp 252–254, DOI 10.1145/2330601.2330661

Slade S, Prinsloo P (2013) Learning analytics: ethical issues and dilemmas. American Behavioral Scientist 57(10):1510–1529, DOI 10.1177/0002764213479366

Stray V, Brandtzaeg EG, Wivestad VT, Barbala A, Moe NB (2026) Developer productivity with and without GitHub Copilot: A longitudinal mixed-methods case study. In: Proceedings of the 59th Hawaii International Conference on System Sciences (HICSS), 2509.20353

Tenschert J, Lenz R (2016) Supporting knowledge work by speech-act based templates for micro processes. In: Business Process Management Workshops (BPM 2016), Springer, Lecture Notes in Business Information Processing, vol 256, DOI 10.1007/978-3-319-42887-1_7

Tu Y, Xiong Y, Chen W, Brinton CG (2018) A domain-independent text segmentation method for educational course content. In: 2018 IEEE International Conference on Data Mining Workshops, IEEE, pp 320–327, DOI 10.1109/ICDMW.2018.00053

Vella A, Blincoe K (2026) The impact of AI coding assistants on software engineering: A longitudinal study. ArXiv:2605.23135, 2605.23135

Xiao H, Eckert C (2013) Efficient online sequence prediction with side information. In: 2013 IEEE 13th International Conference on Data Mining, IEEE, pp 1235–1240, DOI 10.1109/ICDM.2013.31

Yan L, Sha L, Zhao L, Li Y, Martinez-Maldonado R, Chen G, Li X, Jin Y, Gasevic D (2024) Practical and ethical challenges of large language models in education: A systematic scoping review. British Journal of Educational Technology 55(1):90–112, DOI 10.1111/bjet.13370

Zhang Z, Jiang W, Wang Y, Ye Q, Zhao M, Xu M, Wang M (2021) Discriminative additive scale loss for deep imbalanced classification and embedding. In: 2021 IEEE International Conference on Data Mining, IEEE, pp 936–945, DOI 10.1109/ICDM51629.2021.00105