Improving Learning Analytics in Git-Based Learning Ecosystems

Martti Raavel[1], Mart Laanpere[1] and Hans Põldoja[1]

[1]Tallinn University, Narva mnt 25, 10120 Tallinn, Estonia

Abstract

Version control platforms such as GitHub are widely used in computing education to support industry-aligned assignment, feedback, and collaboration workflows. However, GitHub activity traces often provide instructors with limited pedagogical context for understanding assignment progress, submission status, and feedback loops in a timely manner. This makes it difficult to monitor students' work and provide timely scaffolding. This paper presents an instructor-oriented approach to making GitHub activity pedagogically interpretable while preserving authentic student workflows. The approach combines lightweight contextual enrichment of assignment-related GitHub artifacts with a deterministic rule-based translation layer that maps technical event sequences into an assignment-centered pedagogical state vocabulary. We demonstrate the approach using authentic repository timelines from a programming course. The comparison between raw GitHub traces and enriched pedagogical traces shows that lightweight enrichment substantially improves assignment-level trace attribution and makes feedback cycles and progress states more readily observable. These results suggest that minimal semantic enrichment can support instructor awareness, reduce the interpretive effort required to monitor GitHub-based coursework, and provide a practical foundation for building pedagogically meaningful learning analytics dashboards in Git-based learning ecosystems.

Keywords

Learning analytics, GitHub, trace enrichment, instructor awareness, computing education

1. Introduction

Version Control Systems (VCS) and platforms such as GitHub are now widely used in computing education to support industry-aligned workflows for assignments, collaboration, and feedback. In many courses, GitHub is not only a submission channel but also a workspace in which students receive tasks, develop solutions, request reviews, and interact with instructors in ways that resemble contemporary software development practice [1, 2, 3, 4].

While this alignment with professional practice is pedagogically valuable, it also creates a challenge for instructors. The traces produced by GitHub, such as issue events, commits, pull requests, and review actions, are primarily technical records of platform activity. They describe what happened in GitHub, but often do not directly indicate what that activity means pedagogically. For example, a pull request may signal ongoing work in one course but formal submission in another, while issue closure may indicate completion, acceptance, or merely administrative cleanup depending on local workflow rules. As a result, instructors may struggle to determine which assignments are actively being worked on, which have been submitted, where feedback cycles are occurring, and which students may need support.

This interpretive difficulty becomes particularly important in courses and programs where GitHub-based workflows are used at scale across multiple assignments, instructors, and cohorts. In such settings, instructors need timely and understandable indicators of progress in order to monitor work, coordinate feedback, and intervene when needed. Learning analytics dashboards can support this kind of awareness, but only when the underlying traces carry stable semantics aligned with instructional questions [5]. Similarly, educational process mining depends on explicit case identifiers and domain-relevant attributes in order to reconstruct meaningful process instances from low-level event logs [6]. Although standards such as xAPI support the storage of learning-related statements, they do not solve the prior problem of

Baltic DB&IS 2026 Conference Forum and Doctoral Consortium, 28 June - 1 July 2026, Tartu, Estonia

martti.raavel@tlu.ee (M. Raavel); mart.laanpere@tlu.ee (M. Laanpere); hans.poldoja@tlu.ee (H. Põldoja)

0009-0008-9992-2422 (M. Raavel); 0000-0002-9853-9965 (M. Laanpere); 0000-0002-5960-8358 (H. Põldoja)

how technical platform events should be contextualized so that they can be interpreted pedagogically [7].

This paper addresses that problem by proposing a lightweight, instructor-oriented approach for making GitHub activity pedagogically interpretable. The approach combines minimal contextual enrichment of assignment-related GitHub artifacts with a transparent rule-based translation layer that maps event sequences into assignment-centered pedagogical states suitable for instructor dashboards and learning analytics.

We address the following research questions:

RQ1: What pedagogical ambiguities in raw GitHub activity traces make assignment progress and feedback processes difficult for instructors to interpret?

RQ2: How can lightweight contextual enrichment and rule-based interpretation make GitHub-based assignment workflows more understandable for instructors without disrupting authentic student workflows?

RQ3: What instructor-relevant insights about assignment progress and feedback loops become visible when enriched pedagogical traces are used instead of raw GitHub logs?

The contributions of this paper are: (1) an instructor-oriented model for lightweight contextual enrichment of GitHub-based assignment activity that enables assignment-centered interpretation of technical traces; and (2) a transparent rule-based translation approach that converts enriched event sequences into an auditable pedagogical state vocabulary supporting instructor awareness, dashboard design, and downstream learning analytics.

2. Theoretical background

This paper treats GitHub activity as potentially meaningful learning evidence, while recognizing that platform logs are often ambiguous unless interpreted in the instructional context that produced them. The theoretical background therefore motivates *minimal contextual enrichment* as a design requirement for instructor-facing interpretability rather than as additional instrumentation for students.

Activity Theory (AT) explains why actions become interpretable only when related to the object of activity and the mediating conditions under which the activity is carried out, including rules, roles, and division of labor [8]. In GitHub-based coursework, identical event types such as issue closure or review request can have different pedagogical meanings depending on course rules and actor role. This motivates explicit assignment context and role resolution to support stable interpretation across courses.

Instrumental Genesis (IG) describes how artifacts become instruments through the intertwined processes of instrumentation, in which the artifact shapes activity through its constraints and affordances, and instrumentalization, in which users develop conventions and adaptations for their purposes [9]. Applied to GitHub, learners may initially produce partial or improvised workflows and later stabilize conventions. This motivates interpreting traces relative to an expected workflow structure, such as submission strategy, through transparent and auditable rules.

Knowledge Maturation (KM) perspectives frame learning as progressive refinement and stabilization of knowledge objects and practices in sociotechnical settings [10]. In GitHub-based coursework, issues, pull requests, and reviewed code can therefore be treated as evolving work artifacts. This supports assignment-centered state models and linkage conventions that make task progression and feedback cycles more interpretable.

Finally, classroom orchestration research emphasizes instructors' need for timely awareness and intervention without disrupting authentic practice [11]. This motivates translating enriched event streams into assignment-centered pedagogical states suitable for monitoring progress and feedback loops at scale.

Taken together, these perspectives justify the proposed approach. Activity Theory explains why pedagogical meaning depends on context, rules, and roles; Instrumental Genesis explains why GitHub workflows are gradually appropriated and stabilized in practice; Knowledge Maturation frames assignment artifacts as evolving objects of work; and classroom orchestration highlights the need for actionable instructor awareness. Together, they support the design of a lightweight enrichment and rule-based interpretation approach that makes GitHub activity more pedagogically interpretable for instructors.

3. Related work

Git and GitHub have been adopted in computing education as platforms for distributing work, coordinating collaboration, and supporting industry-aligned workflows. Prior studies report both benefits and recurring onboarding challenges: students often experience confusion around version control concepts and conventions, while instructors vary in the tooling and scaffolding they provide [1, 2, 12]. GitHub has also been characterized as an educational coordination space where issues, pull requests, and discussions mediate collaboration and communication beyond simple submission [3, 4].

A smaller but relevant body of work has explored the use of GitHub and Git-based traces for educational monitoring, contribution analysis, and instructor support. Prior studies have examined tools and dashboards for analyzing student teamwork, attributing work in programming teams, and monitoring project progress through repository activity data [13, 14, 15]. These studies demonstrate the value of mining Git-based activity for educational purposes, especially in team-based software engineering contexts. However, they typically emphasize contribution metrics, teamwork visualization, or ad hoc trace analysis rather than the minimal contextual enrichment needed to make raw GitHub events pedagogically interpretable at the assignment level across varying workflows.

Learning analytics research argues that instructor-facing dashboards are useful only when indicators are grounded in stable semantics aligned with instructional decisions [5]. Educational process mining makes a related point from an event-log perspective: interpretable models typically require case identifiers and domain attributes that connect low-level events to meaningful process instances [6]. In GitHub-based coursework, raw event types often under-specify intent, which complicates reconstructing assignment progress and feedback loops at scale.

Interoperability standards such as the Experience API (xAPI) provide a structured format for recording learning-relevant statements in a Learning Record Store (LRS), enabling downstream analysis and dashboarding across tools [7]. However, such standards do not resolve the upstream design problem of deciding which contextual information must be captured so that platform events can be interpreted consistently in pedagogical terms.

Overall, existing work demonstrates the feasibility and value of GitHub-based workflows and trace mining in educational settings, but it rarely specifies the *minimal* enrichment and linkage conventions needed to deterministically interpret GitHub activity as assignment-centered progress and feedback states while preserving low-friction, industry-aligned student workflows. This paper addresses that gap by proposing an instructor-oriented enrichment model and a transparent rule-based translation layer from enriched GitHub events to an auditable pedagogical state vocabulary.

4. Context and settings

This work takes place in the Applied CS program at Tallinn University Haapsalu College (Estonia), a professionally oriented curriculum with a heterogeneous student population that includes adult learners alongside students entering directly from secondary education. Many learners combine studies with employment and family responsibilities, which increases the importance of timely instructor awareness and targeted pedagogical support when assignment progress becomes difficult to interpret from platform activity alone.

4.1. Git-based learning ecosystem

Git and GitHub are used program-wide to distribute learning resources and to enact assignment, submission, and feedback workflows using industry-aligned artifacts such as issues, branches, pull requests, and reviews. Adoption and conventions vary across courses and instructors as part of an intentional onboarding strategy. While this supports authentic professional practice, it also creates interpretation ambiguity when analyzing platform logs, because identical technical events may have different pedagogical meanings depending on course rules, submission strategy, and actor roles.

In Fall 2025, the program-wide GitHub environment spanned six courses taught by three different instructors, with each course enrolling approximately 17–23 students. This multi-instructor, multi-course use of GitHub reinforces the need for trace semantics that remain pedagogically interpretable across varying conventions, workflows, and levels of instructor adoption.

The ecosystem includes an instructor-facing orchestration tool that operates alongside students' GitHub-native workflows. Students complete their work directly in GitHub and receive notifications via Discord, while instructors use the orchestration tool to provision repositories, post assignment issues, and monitor progress across repositories. Identity, roster, and role mappings are managed centrally via Keycloak [16]. Figure 1 summarizes the main components and integrations among Keycloak, the orchestration tool, GitHub repositories, and Discord.

The broader ecosystem architecture and program-level design decisions, including the iterative development of the orchestration tool through multi-year experimentation across courses and cohorts, are reported in prior work [17]. The present paper focuses on how activity traces from this ecosystem can be made pedagogically interpretable for instructor awareness through lightweight enrichment and assignment-centered state reconstruction.

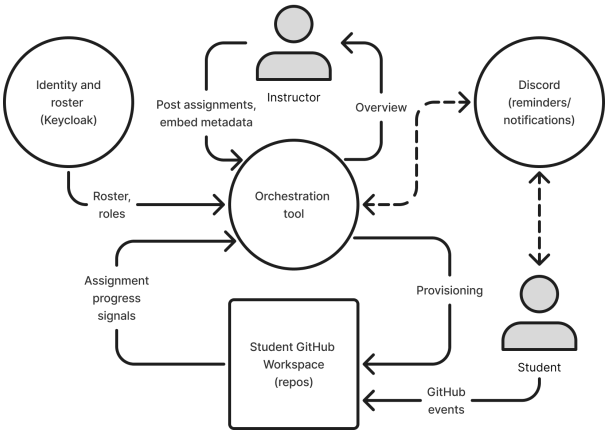

Figure 1: Instructor-facing orchestration environment and its integrations with identity and roster management, student GitHub workspaces, and Discord notifications.

5. Method

This study uses a design-and-demonstrate approach to examine how GitHub activity can be made more pedagogically interpretable for instructors through lightweight contextual enrichment and rule-based interpretation. The purpose is not to evaluate GitHub as a technical platform in itself, but to compare how the same repository history appears from two perspectives: as raw GitHub activity and as assignment-centered pedagogical traces. The method therefore focuses on whether enrichment and translation make assignment progress, submission, and feedback loops easier to reconstruct from an instructor perspective.

The method consists of four core processing steps: (1) extract an event timeline from GitHub, (2) apply a minimal enrichment scheme and linkage conventions anchored to assignment issues, (3) translate enriched events into an assignment-centered pedagogical state model using deterministic rules, and (4)

compare raw and enriched representations with regard to instructor-facing interpretability. Figure 2 also shows how the resulting states support instructor-facing monitoring and decision support.

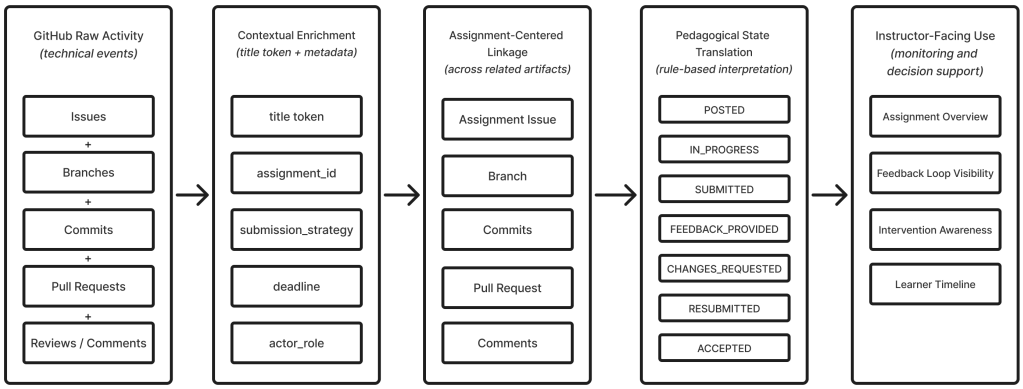

Figure 2: Overview of the proposed approach: raw GitHub activity is enriched with lightweight assignment context, linked across related artifacts, translated into pedagogical states, and made available for instructor-facing monitoring and decision support.

5.1. Demonstration context and data

The demonstration focuses on a Programming II course with second-year students. The observation window spans 2025-09-04 to 2025-12-09. The analysis uses an authentic event timeline from one student repository, selected because it exhibits workflow patterns representative of those observed across students in the course. The dataset contains 203 event rows derived from issues, commits, pull requests, reviews, merges, and reactions. The instructor posted one provisioning issue and seven assignment issues; six used pull request-based submission and one used comment-based completion.

The use of a single repository does not aim at statistical generalization. Rather, it provides an analytically tractable case for demonstrating how the proposed approach changes the interpretability of assignment-related activity for instructors.

5.2. Event sources and extraction

For offline replay during development, repository artifacts and timelines were collected using a custom extractor based on the GitHub APIs. The extracted event table includes timestamp, event type, actor, artifact identifiers, and relevant payload fields such as review state, labels, and assignees. Extracted sequences and counts were cross-checked against the GitHub web interface. Although this paper uses offline replay, the same translation logic is intended for online event streams.

5.3. Role resolution

Pedagogical interpretation is role-sensitive because the same platform action may have different meanings depending on whether it was performed by a student, instructor, or automation. Events are therefore annotated with actor role using a centralized roster and role mapping. In the broader ecosystem, identity and roster management are handled via Keycloak; for the demonstration, events include an `is_instructor` flag derived from this mapping.

5.4. Assignment grounding and linkage

The unit of interpretation is the assignment, grounded in an assignment issue that serves as the canonical task object. Because repository-local issue numbers are not stable across student repositories, assignment episodes are identified using a cohort-wide identifier token generated at posting time by the orchestration tool and embedded in the issue title, for example `[PRG2-F25-A02]`.

Linkage is achieved by propagating the same identifier through GitHub-native artifacts using lightweight naming and referencing conventions. The identifier appears in branch names, can be inherited into pull request titles or descriptions, and may also be connected through issue references or closing relations. During processing, `assignment_id` is resolved primarily from the issue-title token and its propagated appearances in related artifacts. If a structured metadata block in the issue body also includes an identifier field, it is treated as an optional machine-readable fallback. This enables deterministic issue-to-branch-to-pull-request linkage for assignment-centered aggregation without heuristic matching.

5.5. Minimal enrichment scheme

Two streams are prepared for comparison. The first is a raw stream, in which events are represented only through GitHub-native event types and payload fields. The second is an enriched stream, in which events are augmented with the minimal assignment semantics and role context needed for pedagogical interpretation.

The enrichment includes: (1) `assignment_id`; (2) submission strategy, distinguishing pull request-based and comment-based assignments; (3) deadline, when available; and (4) actor role, distinguishing between student, instructor, and automation. The first three elements are embedded in structured assignment metadata, whereas actor role is derived externally from the identity and roster management system based on the event actor. Figure 3 shows an illustrative example of the issue-embedded metadata used to represent assignment semantics.

```
<!-- pedagogy-metadata
{
  "schema_version": "1.0",
  "submission_strategy": "pull_request",
  "deadline": "2025-10-01T23:59:00+02:00",
  "assignment_id": "PRG2-F25-A02"
}
-->
```

Figure 3: Illustrative issue-embedded JSON metadata for assignment semantics with optional linkage fallback (truncated).

Additional fields such as requiredness and learning-outcome references are supported by the metadata schema but treated as optional in this paper.

5.6. Replay and rule-based translation to pedagogical states

Events are sorted by timestamp and replayed in chronological order within each assignment episode. When multiple events share the same timestamp, a stable tie-breaker is applied so that identical inputs yield identical derived states.

The enriched stream is translated into assignment-centered pedagogical states using an ordered rule set. The state vocabulary is: POSTED, IN_PROGRESS, SUBMITTED, FEEDBACK_PROVIDED, CHANGES_REQUESTED, RESUBMITTED, ACCEPTED.

The purpose of this translation is to express GitHub activity in terms that are more directly meaningful for instructor monitoring. Each event is evaluated using its type, relevant event-specific payload fields

(such as review state, label changes, assignee changes, and merge status), actor role, assignment metadata, and linkage. When a rule matches, it emits a state transition and records provenance.

Operational definitions used in the demonstration are as follows:

- POSTED: instructor creates an assignment issue with parsable submission metadata.
- IN_PROGRESS: first credible student work signal linked to the assignment.
- SUBMITTED: explicit submission handoff, operationalized as review request for pull request-based assignments or designated submission signaling for comment-based assignments.
- FEEDBACK_PROVIDED: instructor feedback event.
- CHANGES_REQUESTED: instructor revision-request signal.
- RESUBMITTED: renewed submission signal after CHANGES_REQUESTED.
- ACCEPTED: instructor acceptance signal through merge or issue closure, depending on submission strategy.

5.7. Comparison approach

To examine the effect of enrichment and translation on instructor-facing interpretability, the analysis contrasts two representations derived from the same repository history: (1) a representation over GitHub-native event types and (2) a representation over translated pedagogical states. The comparison focuses on whether assignment episodes, submission handoffs, and feedback loops can be reconstructed in a way that is understandable without relying on tacit course-specific conventions.

The comparison is interpretive rather than predictive. It does not attempt to measure learning outcomes or model student performance. Instead, it examines whether the enriched representation provides a more pedagogically meaningful account of assignment activity for instructor awareness and monitoring.

6. Results

This section reports the outcomes of applying the proposed enrichment and rule-based translation to an authentic Programming II repository timeline. The results are organized around RQ1 to RQ3 and focus on how the enriched representation changes the interpretability of GitHub-based assignment activity from an instructor perspective.

6.1. RQ1: Pedagogical ambiguities in raw GitHub traces

The raw GitHub timeline revealed several pedagogical ambiguities that make assignment progress difficult for instructors to interpret, especially when events cannot be reliably attributed to a specific assignment episode. The enrichment scheme addressed these ambiguities by making assignment context and cross-artifact linkage machine-readable at event time. In the raw timeline, only issue-scoped events were directly attributable to a specific assignment issue without additional assumptions, because commits and pull requests were not deterministically linked to the originating assignment. In the demonstration dataset, issue events accounted for 62 of 203 events (30.5%). After enrichment, 184 of 203 events (90.6%) carried an `assignment_id` and could be deterministically assigned to an assignment episode.

Enrichment also reduced the semantic complexity that instructors would need to interpret during monitoring. The raw stream contained 14 distinct GitHub event types, whereas the translated representation used 7 pedagogical states. Table 1 summarizes these effects.

For instructor awareness, this means that a much larger share of observed activity can be connected to specific assignments rather than remaining as disconnected technical events. It also means that monitoring can shift from interpreting many low-level GitHub actions to following a smaller set of pedagogically meaningful progress states.

Table 1
Trace attribution and semantic complexity in the raw versus enriched representations

	Raw stream	Enriched stream
Total events	203	203
Distinct GitHub event types	14	14
Directly assignment-attributable events	62 (30.5%)	184 (90.6%)
Pedagogical state vocabulary size	n/a	7

6.2. RQ2: Making GitHub-based workflows more interpretable through enrichment

The proposed enrichment was embedded into existing GitHub-based assignment workflows by grounding each assignment in an issue and reusing a stable `assignment_id` across related artifacts. This made assignment activity more interpretable for instructors while preserving students' ordinary GitHub workflow patterns. In the demonstration repository, one issue represented repository provisioning and seven issues represented assignments, of which six used pull request-based submission and one used comment-based completion. For pull request-based assignments, submission was operationalized as an explicit review-request event so that instructor notification aligned with common industry practice.

Coverage by artifact type shows that issue and pull-request events were fully linkable in the dataset, while linkage gaps occurred only for commits. Notably, 19 commit events (18.4% of commits) lacked the linkage identifier. Manual inspection indicated that these commits were pushed directly to the `main` branch, primarily during in-class lessons or as follow-up changes after a pull request had already been merged, where students did not operate within an assignment-linked branch.

This suggests that the remaining ambiguity is not mainly a limitation of issue and pull-request linkage, but rather a question of how ambient work, that is, background activity not explicitly tied to a specific assignment episode, is represented in the workflow. Two complementary mitigations are possible: (1) enforce a "no direct commits to `main`" convention by requiring branches even during lessons, using a lightweight lesson token such as `[LESSON-05]` or `[PRG2-F25-L05]`; and (2) when direct commits are pedagogically acceptable, treat them explicitly as a separate work stream, for example LESSON_WORK or POST_MERGE_FIX, rather than attempting to attribute them to an assignment episode.

From an instructor perspective, these results indicate that the proposed enrichment can be embedded in ordinary GitHub workflows with relatively little disruption while still preserving high assignment-level traceability. The remaining ambiguity is concentrated in a specific and recognizable class of activity, which makes it possible to refine workflow conventions without redesigning the overall learning environment.

6.3. RQ3: Instructor-relevant insights made visible by enriched pedagogical traces

Rule-based translation made assignment progress and feedback loops visible in a form that is more directly meaningful for instructor awareness. Across the seven assignments, the replay yielded six assignments reaching ACCEPTED within the observation window, of which two required additional changes. One assignment remained in IN_PROGRESS without a submission signal by the end of the observation window.

State-change transitions, excluding transitions where the pedagogical state did not change, indicate that the dominant progression followed the expected lifecycle from POSTED to IN_PROGRESS, then to SUBMITTED, FEEDBACK_PROVIDED, and ACCEPTED. The replay also surfaced two non-monotonic transitions from SUBMITTED back to IN_PROGRESS, caused by work signals arriving after the submission handoff. These cases suggest the need for stronger monotonicity constraints, for example disallowing IN_PROGRESS after SUBMITTED unless preceded by CHANGES_REQUESTED. Table 2 summarizes the observed state-change transitions for the seven assignments.

For instructor awareness, the translated representation makes it possible to distinguish ongoing work, submission, revision, feedback, and acceptance without relying on tacit knowledge of how a particular course uses GitHub events. In practical terms, this means that a teacher can more readily see

which assignments are waiting for review, which have entered a revision cycle, and which have not yet reached a submission handoff. At the same time, the observed linkage gaps and non-monotonic transitions provide concrete targets for improving workflow conventions and rule guards before live deployment.

Table 2
Observed state-change transitions for the seven assignments, excluding transitions where the pedagogical state did not change

Transition	Count
POSTED → IN_PROGRESS	7
IN_PROGRESS → SUBMITTED	6
SUBMITTED → FEEDBACK_PROVIDED	6
FEEDBACK_PROVIDED → CHANGES_REQUESTED	2
CHANGES_REQUESTED → RESUBMITTED	2
RESUBMITTED → FEEDBACK_PROVIDED	2
FEEDBACK_PROVIDED → ACCEPTED	6
SUBMITTED → IN_PROGRESS (non-monotonic)	2

7. Discussion

The findings suggest that improving instructor awareness in GitHub-based coursework is not primarily a matter of adding more dashboards or more metrics, but of making existing activity traces pedagogically interpretable. GitHub logs are detailed, yet their default semantics are technical rather than educational. Without stable assignment linkage, explicit submission semantics, and role-aware interpretation, the same platform event can carry different meanings across courses and instructors. The proposed minimal enrichment therefore targets the smallest set of additions needed to make assignment activity understandable for instructor monitoring while preserving authentic student workflows.

7.1. Minimal enrichment as a foundation for instructor awareness

A central implication is that minimal enrichment can support scalable instructor awareness with low additional overhead for students. By grounding each assignment in an issue and linking related artifacts such as branches, pull requests, and commits to the same assignment identifier, activity can be reconstructed as assignment episodes rather than as disconnected technical events. This is visible in the coverage comparison: only 30.5% of raw events were directly attributable to an assignment episode, whereas 90.6% were attributable after enrichment (Table 1). The main gain is therefore not the collection of more data, but the transformation of existing activity into a form that is more interpretable for instructional monitoring.

Once enriched events are translated into a compact pedagogical state vocabulary, instructors can monitor work at the level where pedagogical action occurs. Instead of interpreting many low-level GitHub actions, they can follow states such as IN_PROGRESS, SUBMITTED, CHANGES_REQUESTED, and ACCEPTED. The observed trajectories show both a dominant progression path and revision cycles, which are precisely the kinds of patterns that matter when deciding where support, feedback, or follow-up is needed. The resulting pedagogical states also provide a clearer conceptual foundation for instructor dashboards, because they expose assignment progress and feedback cycles directly rather than requiring instructors to interpret low-level GitHub events manually.

Even in this small demonstration, the non-monotonic transitions show why an explicit interpretation layer is useful. Without such a layer, instructors may draw misleading conclusions from technically correct but pedagogically ambiguous event sequences. In this sense, the contribution is not only technical trace enrichment, but also support for pedagogical orchestration and instructor sense-making in GitHub-based learning environments.

This design also supports pedagogical consistency across courses and instructors. If assignment issues are posted with common linkage conventions and explicit submission-strategy metadata, instructors who differ in their day-to-day GitHub practices can still produce comparable pedagogical traces. Enrichment therefore functions not only as a traceability mechanism for analytics, but also as a lightweight pedagogical standardization mechanism across courses.

7.2. Design implications: linkage and submission semantics

The results highlight that linkage is a first-class requirement for instructor-facing interpretation. The enriched stream achieved near-complete linkage coverage for issues, pull requests, reviews, merges, and issue comments, while commits remained the primary source of ambiguity. This is unsurprising, because commits are both frequent and likely to occur outside clearly bounded assignment episodes unless workflow conventions are followed consistently.

These findings support keeping linkage conventions visible, simple, and aligned with authentic GitHub use. An instructor-generated assignment identifier embedded in the issue title can propagate naturally to branch names and then to pull request titles or descriptions. This creates a robust and human-readable mechanism for assignment attribution that remains compatible with common professional workflows and avoids dependence on repository-local issue numbers, which are not stable across student repositories.

The demonstration also reinforces that submission should be treated as a pedagogical handoff rather than as a single technical event independent of context. For pull request-based assignments, a review request is a meaningful submission signal because it indicates that the student is explicitly handing work over for instructor attention. For comment-based assignments, a designated signal such as applying a submission label and assigning the instructor serves a similar pedagogical function. Making submission strategy explicit in issue metadata therefore reduces ambiguity both for the translation layer and for instructors who need to interpret student progress consistently.

From a theoretical perspective, these findings align with Activity Theory and classroom orchestration perspectives introduced earlier in the paper. The pedagogical meaning of GitHub events depends on rules, roles, and instructional context rather than on the technical event type alone. The enrichment approach therefore acts as a bridge between low-level platform activity and the pedagogical processes instructors need to monitor and regulate.

The contribution of this paper is intentionally limited to what is needed for reliable assignment state reconstruction and near-real-time instructor awareness. This minimal layer is not an endpoint, but a foundation. Once assignment episodes can be identified and pedagogical states reconstructed consistently, richer forms of interpretation can be built incrementally, including links to learning outcomes, distinctions between mandatory and optional work, and more nuanced representations of revision intent or work quality. These additions can extend the same interpretation framework without changing the student-facing workflow fundamentals.

7.3. Limitations

Consistent with the scope of the research questions, this study focuses on the interpretability of activity traces rather than on direct measurement of learning outcomes. The demonstration uses one course and one student repository selected as representative, so the findings support feasibility and plausibility rather than statistical generalization across students, courses, or instructors.

The approach also depends on the quality of linkage conventions and metadata. If students omit `assignment_id` references in branches or pull requests, or if role mappings are incomplete, attribution errors can occur. In addition, the rule-based translation layer reflects local policy choices, such as what counts as submission or acceptance, which means that portability requires adjustment and validation in new contexts.

Finally, the demonstration relies on offline API-based extraction and chronological replay. A live webhook-based implementation will need to handle duplicates, delayed or out-of-order delivery, and

platform side effects such as merges that automatically close issues. These are implementation challenges that must be addressed before operational deployment, although they do not change the conceptual contribution of the pedagogical enrichment approach itself.

7.4. Future work

Future work will focus on deploying the approach in live courses and examining how enriched pedagogical traces support instructor awareness and pedagogical decision making in practice. In addition to implementing the webhook-driven pipeline and downstream learning analytics infrastructure, an important next step is evaluating how instructors use enriched representations when monitoring progress, identifying bottlenecks, prioritizing feedback, and responding to students.

We also plan to evaluate the approach across multiple instructors and courses through reconstruction and decision-making tasks, for example by comparing time-to-awareness, consistency of interpretation, and perceived usefulness of raw versus enriched representations. These studies would make it possible to investigate not only the technical correctness of the enrichment approach, but also its pedagogical usefulness in authentic teaching practice.

In parallel, the enrichment scheme can be extended beyond the minimal core through optional metadata fields supporting richer interpretations aligned with Instrumental Genesis and Knowledge Maturation, such as learning-outcome links, artifact types, revision intent, and longer-term developmental patterns in collaborative work practices.

Finally, future work will examine portability by applying the conventions and rule set across additional course settings and refining templates, governance, and onboarding practices that support program-level consistency in Git-based learning ecosystems.

8. Conclusion

GitHub-based assignment and assessment workflows can support industry-aligned software development practices in computing education, but the resulting activity traces are primarily technical and often insufficient for reliable, real-time instructor awareness of assignment progress and feedback processes. This paper introduced a pedagogical trace enrichment approach that combines lightweight contextual enrichment of GitHub artifacts with a deterministic rule-based translation layer for reconstructing assignment-centered pedagogical states.

Using an authentic Programming II repository timeline, the comparison between raw and enriched representations demonstrated how minimal semantic enrichment improves the reconstructability of assignment progress, submission handoffs, and feedback loops without requiring additional student-facing tools or disrupting authentic GitHub workflows. The findings suggest that pedagogically enriched traces can reduce the interpretive effort required from instructors, support more consistent monitoring across courses, and provide a practical foundation for instructor-facing learning analytics dashboards in Git-based learning ecosystems.

Declaration on Generative AI

The author(s) have not used Generative AI tools to adjust the text's language and tone.

During the preparation of this work, the authors used ChatGPT for sentence polishing and rephrasing in order to improve clarity, readability, and language quality. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] L. Haaranen, T. Lehtinen, Teaching Git on the Side: Version Control System as a Course Platform, in: Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science

Education, ACM, Vilnius Lithuania, 2015, pp. 87–92. URL: https://dl.acm.org/doi/10.1145/2729094.2742608. doi:10.1145/2729094.2742608.

[2] V. Isomöttönen, M. Cochez, Challenges and Confusions in Learning Version Control with Git, in: V. Ermolayev, H. C. Mayr, M. Nikitchenko, A. Spivakovsky, G. Zholtkevych (Eds.), Information and Communication Technologies in Education, Research, and Industrial Applications, volume 469, Springer International Publishing, Cham, 2014, pp. 178–193. URL: https://link.springer.com/10.1007/978-3-319-13206-8_9. doi:10.1007/978-3-319-13206-8_9, series Title: Communications in Computer and Information Science.

[3] A. Zagalsky, J. Feliciano, M.-A. Storey, Y. Zhao, W. Wang, The Emergence of GitHub as a Collaborative Platform for Education, in: Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing, ACM, Vancouver BC Canada, 2015, pp. 1906–1917. URL: https://dl.acm.org/doi/10.1145/2675133.2675284. doi:10.1145/2675133.2675284.

[4] J. Feliciano, M.-A. Storey, A. Zagalsky, Student experiences using GitHub in software engineering courses: a case study, in: Proceedings of the 38th International Conference on Software Engineering Companion, ACM, Austin Texas, 2016, pp. 422–431. URL: https://dl.acm.org/doi/10.1145/2889160.2889195. doi:10.1145/2889160.2889195.

[5] K. Verbert, E. Duval, J. Klerkx, S. Govaerts, J. L. Santos, Learning Analytics Dashboard Applications, American Behavioral Scientist 57 (2013) 1500–1509. URL: https://journals.sagepub.com/doi/10.1177/0002764213479363. doi:10.1177/0002764213479363.

[6] A. Bogarín, R. Cerezo, C. Romero, A survey on educational process mining, WIREs Data Mining and Knowledge Discovery 8 (2018) e1230. URL: https://wires.onlinelibrary.wiley.com/doi/10.1002/widm.1230. doi:10.1002/widm.1230.

[7] ADL Initiative, adlnet/xAPI-Spec, 2026. URL: https://github.com/adlnet/xAPI-Spec, original-date: 2012-12-19T16:46:59Z.

[8] Y. Engestrom, Activity theory as a framework for analyzing and redesigning work, Ergonomics 43 (2000) 960–974. URL: https://www.tandfonline.com/doi/full/10.1080/001401300409143. doi:10.1080/001401300409143.

[9] P. Rabardel, From artefact to instrument, Interacting with Computers 15 (2003) 641–645. URL: https://academic.oup.com/iwc/article-lookup/doi/10.1016/S0953-5438(03)00056-0. doi:10.1016/S0953-5438(03)00056-0.

[10] T. Ley, R. Maier, S. Thalmann, L. Waizenegger, K. Pata, A. Ruiz-Calleja, A Knowledge Appropriation Model to Connect Scaffolded Learning and Knowledge Maturation in Workplace Learning Settings, Vocations and Learning 13 (2020) 91–112. URL: http://link.springer.com/10.1007/s12186-019-09231-2. doi:10.1007/s12186-019-09231-2.

[11] P. Dillenbourg, Design for classroom orchestration, Computers & Education 69 (2013) 485–492. URL: https://linkinghub.elsevier.com/retrieve/pii/S0360131513001061. doi:10.1016/j.compedu.2013.04.013.

[12] R. Glassey, Adopting Git/Github within Teaching: A Survey of Tool Support, in: Proceedings of the ACM Conference on Global Computing Education, ACM, Chengdu,Sichuan China, 2019, pp. 143–149. URL: https://dl.acm.org/doi/10.1145/3300115.3309518. doi:10.1145/3300115.3309518.

[13] J. J. Sandee, E. Aivaloglou, GitCanary: A Tool for Analyzing Student Contributions in Group Programming Assignments, in: Koli Calling '20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research, ACM, Koli Finland, 2020, pp. 1–2. URL: https://dl.acm.org/doi/10.1145/3428029.3428563. doi:10.1145/3428029.3428563.

[14] M. Guttmann, A. Karaka, D. Helic, Attribution of Work in Programming Teams with Git Reporter, in: Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1, ACM, Portland OR USA, 2024, pp. 436–442. URL: https://dl.acm.org/doi/10.1145/3626252.3630785. doi:10.1145/3626252.3630785.

[15] N. Gitinabard, S. Heckman, T. Barnes, C. F. Lynch, Designing a Dashboard for Student Teamwork Analysis, in: Proceedings of the 53rd ACM Technical Symposium on Computer Science Education, 2022, pp. 446–452. URL: http://arxiv.org/abs/2112.04465. doi:10.1145/3478431.3499377, arXiv:2112.04465 [cs].

[16] Keycloak Project, Keycloak documentation, https://www.keycloak.org/documentation, 2026. Accessed: 2026-01-18.

[17] M. Raavel, M. Laanpere, H. Põldoja, Designing a knowledge-building ecosystem for computer science students using version control systems, in: Proceedings of the IEEE Global Engineering Education Conference (EDUCON), 2026. In press.