USING PULL REQUESTS TO MAKE COLLABORATION VISIBLE IN CDIO PROJECT-BASED COURSES

Helga Ingimundardóttir

Industrial Engineering, University of Iceland

ABSTRACT

Assessing collaboration, individual accountability, responsiveness to feedback, and reproducibility remains a persistent challenge in open-ended, team-based project courses. Final deliverables often mask how work evolved, how responsibilities were shared, and how decisions were negotiated. This paper presents an assessment design in which pull requests (PRs), a mechanism for reviewing and integrating changes to shared project work, serve as the central mechanism for making collaborative practices visible and assessable within CDIO-aligned project-based courses. Rather than treating GitHub as a programming topic or tooling exercise, the approach positions PRs as assessment infrastructure that binds technical change to process evidence through review discussions, explicit responses to comments, and documented handover. The implementation spans two intentionally sequenced courses in Industrial Engineering. A second-year course introduces structured collaboration practices through short, skills-focused project cycles. A subsequent advanced course assumes GitHub fluency and organizes all project work within a single persistent repository maintained across the semester. Within this environment, PRs function as the primary unit of feedback, peer review, and evaluation. Students make their reasoning, revisions, and communication visible through routine workflow activity, and unresolved feedback carries forward across project cycles into capstone evaluation, enabling cumulative rather than fragmented assessment. Experience across multiple offerings shows that PR-centered assessment improves the scope and specificity of peer review, strengthens documentation and reproducibility practices, and fosters shared responsibility for evolving team artifacts. Students initially expressed concern about perceived overhead, but review quality, documentation habits, and disciplined handover improved markedly as the workflow became routine. Graduating students later identified PR-based communication and accountability as directly transferable to professional settings. The paper highlights key design decisions, trade-offs, and sequencing choices, offering practical and transferable guidance for educators seeking assessment methods that make collaboration, responsiveness, and reproducibility genuinely visible in CDIO project-based courses.

KEYWORDS

Pull requests, GitHub Classroom, assessment infrastructure, team-based learning, professional competence, CDIO standards: 5, 7, 8, 11

INTRODUCTION AND MOTIVATION

Team-based projects pose persistent assessment challenges: instructors must evaluate not only technical outcomes but also how teams coordinate work, share responsibility, respond to feedback, and hand over results in reproducible, trustworthy ways. In many courses, final deliverables reveal little about how work evolved or how responsibility was shared, motivating assessment approaches that make collaborative processes visible rather than relying solely on end products.

Existing uses of GitHub Classroom in CDIO and computing-education contexts focus on automated grading and assignment distribution (Glassey, 2019), leveraging test frameworks and repository templates to check correctness rather than to support collaborative review. Although richer GitHub features–particularly *pull requests (PRs)*, a mechanism through which proposed changes to a shared codebase/project are reviewed, discussed, and approved before integration–are considered promising for structuring critique and accountability, their pedagogical use in assessment remains limited, despite the broader adoption of GitHub in education. In contrast, Petre and Wilson (2014) emphasize that small, just-in-time reviews embedded in normal workflows surface reasoning and improve reproducibility, and practitioner guidance in scientific computing highlights documentation, version control, and transparent handover as foundational collaborative practices (Wilson et al., 2017).

Throughout this paper, PRs are treated as principal assessment artifacts. A PR couples a focused change set with a clear description of intent, threaded review comments, and documented responses to feedback. Unlike commits or standalone reports, PRs provide sustained visibility into how work evolves through critique and revision and how responsibility is distributed within a team. Relocating substantive communication from transient chat tools into *Issues* (task- and discussion-tracking threads within the repository) and PR threads further strengthens traceability and accountability.

The aim of this work is not the adoption of a particular platform, but the development of an assessment architecture that foregrounds collaboration, responsiveness to feedback, and reproducibility in project-based courses. This paper contributes such an architecture by positioning PRs as the core assessment unit and demonstrating its implementation across two intentionally sequenced courses, showing how PR-centered workflows create visible, attributable evidence of learning and support coherent, cumulative evaluation in CDIO-aligned curricula.

COURSE CONTEXT AND CURRICULAR PLACEMENT

The implementation spans two intentionally sequenced courses serving Industrial Engineering students, with regular enrollment from adjacent programs.

Information Engineering (IE) is a second-year mandatory undergraduate course built around short, skills-focused project cycles that consolidate foundational competencies. GitHub collaboration practices (Issues for task tracking, branches for isolated work, and PRs for review and handover) are explicitly introduced and scaffolded. Each cycle uses a separate repository to reset expectations, isolate errors, and manage cognitive load, emphasizing skill development and guided practice over long-term continuity. The course typically enrolls 20–25 students organized into small teams of 3–4 (6–7 teams).

Business Intelligence (BI) is a third-year undergraduate/first-year master's elective with larger, data-intensive projects organized in thematic cycles culminating in a final deliverable. GitHub is the primary collaboration environment, and proficiency is assumed. Assessment centers on *iterative, PR-based workflows* within a *single persistent team repository* used across the semester, enabling teams to revisit, resolve, and build upon feedback across successive PRs. The course typically enrolls 10–15 students working in two larger teams. Further detail on the BI course design appears in Ingimundardóttir (2026).

IMPLEMENTATION HISTORY AND IDENTIFIED LIMITATIONS

Early offerings of the BI course used one repository per cycle, with each cycle assessed independently. Although PRs supported review within cycles, work traces were fragmented across repositories: discussions, diffs, documentation, and decisions were split, making it difficult to audit how feedback was addressed. Instructor comments became effectively non-binding once a repository closed, and collaboration rules had to be repeatedly configured.

A further issue emerged in practice. In the first iteration of the course, the initial project cycle was consistently the weakest: teams struggled not only with problem framing and analytic skills but also with learning GitHub workflows from scratch. Students were simultaneously acquiring technical content, collaboration conventions, and version-control practices, and the combined cognitive load slowed early progress. Subsequent cycles improved markedly once basic GitHub fluency was established, indicating that tool onboarding was misaligned with the project's complexity.

This observation motivated a structural change in the curriculum. GitHub onboarding was moved to the prerequisite IE course, where collaboration workflows could be introduced through short, lower-stakes cycles. With these competencies in place, the BI course could assume fluency and shift attention to deeper analytic work and more substantive review practices. BI therefore adopted a single persistent repository with cumulative PR continuity, ensuring that feedback, decisions, and documentation remained visible and actionable across the semester. This design aligns with CDIO program practices that emphasize intentional sequencing and coherent integration of learning outcomes across courses (Malmqvist et al., 2006).

DESIGN RATIONALE: PULL REQUESTS AS ASSESSMENT INFRASTRUCTURE

The design objective is to make collaboration, responsiveness to feedback, and responsibility for shared artifacts *visible and assessable through normal project activity*. Prior CDIO work underscores that documentation habits only persist when they are explicitly structured and routinely reinforced; for example, Junaid et al. (2018) show that logbook keeping supports learning and traceability only when embedded in deliberate practice with clear expectations. We extend this principle from individual records to collaborative artifacts: when communication, critique, and decision-making occur within shared, version-controlled workflows, they become durable and auditable in ways that isolated deliverables do not.

Within this framing, PRs serve as assessment units that bind *technical change* to *collaborative process*. A PR couples a focused change set with rationale and scope, line-level review, threaded discussion, explicit responses to comments, and a visible merge or revise decision.

Because these traces are produced *in situ*–as part of the team's routine workflow–they provide integrated, assessable evidence of reasoning, critique, revision, and handover, aligning with recommendations on just-in-time review and reproducible practice (Petre & Wilson, 2014; Wilson et al., 2017).

Infrastructure context: GitHub Classroom

GitHub Classroom provisions standardized repositories from a common template and centralizes instructor-student interaction within Issues, pull request reviews, and feedback PRs (GitHub Inc., 2026, for documentation). Bennedsen et al. (2022) describe how GitHub Classroom has primarily been used to organize assignment distribution and feedback in repositories. More broadly, Glassey (2019) surveys Git/GitHub use in computing education and highlight its role in assignment management, batch cloning, and artifact-based review.

PRs as assessment units

PRs bind *technical change* to *collaborative process*. A PR couples a focused change set with rationale and scope, line-level review, threaded discussion, explicit responses to comments, and a visible merge or revise decision. Commits rarely expose intent, and standalone reports detach explanation from the evolving artifact. PRs provide integrated, assessable evidence of reasoning, critique, revision, and decision making (Glassey, 2019; Petre & Wilson, 2014).

ASSESSMENT ARCHITECTURE AND IMPLEMENTATION PRACTICES

Figure 1 illustrates the interaction structure of a typical project cycle. Students begin from a shared repository template and work on commits within separate branches. All changes to main must be integrated through PRs; direct commits to main are not permitted. Responsibility for integration is therefore distributed: the student responsible for a branch must request reviews from teammates, where peers are responsible for evaluating, commenting on, and approving changes before merging. One contribution is developed as a dependent branch and merged back into the core work stream, while another proceeds through a PR phase involving iterative revision before being merged into main. After the parallel feature is merged, the updated state of main is synchronized back into the core work stream, ensuring that subsequent integration reflects the current shared state. The final integrated result on main is then subject to instructor feedback.

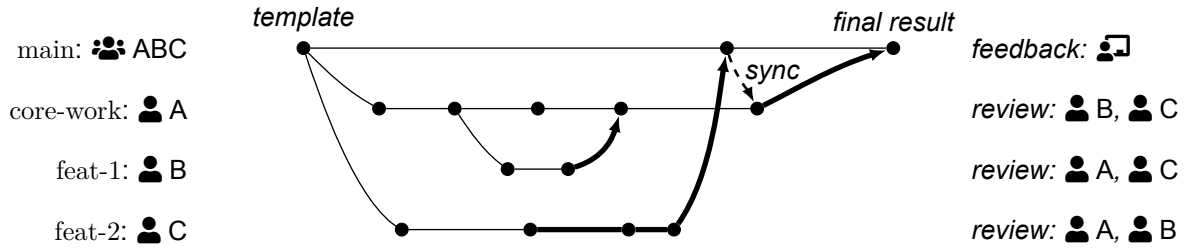

Figure 1. PR-centered interaction workflow for one project cycle. Students work on separate branches and integrate changes through reviewed pull requests. The PR phase is highlighted by thicker edges, indicating reviewed integration rather than independent work.

Assessable behaviors and required practices

The assessment architecture makes behaviors explicit and reviewable. Students must:

1. Submit focused PRs with clear descriptions and linked Issues when appropriate.
2. Provide at least two *substantive* peer reviews before merge.
3. Respond explicitly to peer and instructor comments (no silent fixes or private replies).
4. Merge only after approvals, documenting any deferrals as follow-up Issues.
5. Update README and run instructions when execution, data, or interpretation changes.
6. Participate visibly through authorship, co-authorship, and review.

These practices mirror industry norms, where PRs serve as the locus of communication, accountability, and reproducible handover (Petre & Wilson, 2014). They also reflect documentation-first workflows advocated in scientific computing (Wilson et al., 2017).

Assessment efficiency and reproducibility

Repository-based workflows keep all submission files *in situ* within a standard project structure, eliminating ad hoc packaging (e.g. ZIPs) and ensuring that code runs directly in that structure, with shared datasets explicitly included or referenced. This reduces non-pedagogical overhead (such as fixing paths, locating files, rebuilding contexts) and redirects effort toward evaluating method, documentation, and responsiveness. The repository becomes a stable execution and review environment, where diffs, run instructions, and documentation are co-located, improving reproducibility and assessment consistency.

Sequencing across courses

IE introduces collaboration conventions through short, skills-focused cycles. BI assumes fluency and leverages a *single persistent repository* to support cumulative iteration, sustained traceability, and efficient capstone evaluation. This sequencing shifts instructional effort from tooling toward collaboration quality.

ASSESSMENT FOCUS: REPRODUCIBILITY, HANDOVER, AND RESPONSIVENESS

Reproducibility. Documentation is maintained within the repository, especially the evolving README. Contributions that affect execution or interpretation must include run instructions, data-dependency descriptions, and concise summaries of analytical steps. Because documentation is reviewed through PRs, reproducibility is treated as a routine responsibility rather than a final-stage check.

Handover. Each PR functions as an explicit handover. Authors describe intent and scope, link Issues, and explain rationale at the moment of integration, making decisions and reasoning traceable.

Responsiveness. Students must engage directly in PR discussions to answer questions, clarify assumptions, and revise work where appropriate. Questions asked in threads are treated as positive evidence of engagement and shared responsibility. The depth of review and the explicitness of responses provide observable indicators of individual learning outcomes.

Substantive peer review is characterized by specific, actionable comments that address correctness, clarity, or design decisions, and by engagement in follow-up discussion when issues are raised. Superficial approvals without critique are not considered sufficient evidence of participation.

These dimensions operationalize professional competence in observable ways and align with formative, process-oriented assessment approaches: reproducibility through documentation practices, handover through structured PR descriptions, and responsiveness through engagement in review discussions, grounding assessment in observable evidence of attainment (Biggs & Tang, 2015). Taken together, they reinforce collective accountability and strengthen system-level responsibility. In earlier offerings of the course (and in comparable project-based settings) students frequently approached assignments as partitioned tasks, completing only their designated component while treating others' work as opaque. When later stages depended on prior work, students often expressed uncertainty about foundational steps, sometimes stating candidly that "someone else handled that part." This divide-and-conquer pattern is well-documented in team-based learning research, where strict task partitioning leads to superficial engagement and enables students to contribute minimally while relying on others' understanding (Börjesson et al., 2006).

A parallel dynamic is observed in engineering education more broadly: engineering students often delay engagement with conceptually demanding tasks until temporal pressure forces them to confront prior material, using procrastination as a coping mechanism that postpones understanding until "just-in-time" moments such as exams or later project milestones (Kim & Nembhard, 2019; Wilhelm & Nijman, 2023). These patterns reinforce the need for assessment structures that make intermediate understanding visible, distributed, and accountable throughout the project timeline rather than concentrated at high-stakes endpoints.

The assessment architecture makes shared responsibility explicit and ties evaluation to engagement with the evolving system, not isolated subtasks. PR authorship and co-authorship, review quality and documented follow-up provide attributable evidence of individual learning within team artifacts. These traces show whether students scrutinize and improve peers' work; superficial reviews or unexamined merges are visible, while substantive threads capture reasoning, error correction, and negotiated interpretation. Instructors gain actionable evidence of individual attainment and free-riding is deterred as weak collaboration surfaces in the PR history and impacts team performance.

Ultimately, the approach helps students see that effective teamwork requires shared understanding of system integrity, not merely completion of personal fragments. It also gives instructors visibility into misconceptions and reasoning processes, enabling targeted adjustments and coherent support across project cycles.

FEEDBACK CARRY-FORWARD AND CAPSTONE RE-EVALUATION

Project work in BI is organized in thematic cycles culminating in a capstone deliverable. Instructor feedback is delivered through PR reviews and is not closed at cycle boundaries. Unresolved comments automatically *carry forward*. Capstone evaluation checks whether previously identified issues have been resolved; if not, the corresponding assessment outcome remains unchanged. This makes feedback binding without requiring resubmission and enables efficient

Table 1. Rubric used to assess GitHub collaboration practices in the IE course

	Excellent	Good	Insufficient
GitHub usage	Branches are well organized with logical names tied to features. Commits are coherent and clearly describe changes. Documentation is complete and provides clear instructions.	Branches are used but naming or scope is sometimes inconsistent. Commits are mostly coherent with occasional issues. Documentation is present but incomplete.	Branches are absent or unclear. Commits are inconsistent with little structure. Documentation is unclear or missing.
Teamwork	All team members actively contribute. Contributions and responsibilities are distributed across the team. Decisions are shared through review and discussion.	Most members participate, but contributions are uneven. One or two members lead development, while others contribute through reviews or comments.	Work is largely carried out by one individual. Participation is minimal or uneven across the team.

summative assessment, as instructors can verify changes through diffs rather than rereviewing entire codebases. Early cycles may include exploratory or partial solutions, provided limitations are acknowledged and tracked. Capstone evaluation rewards teams that integrate feedback and deliver coherent, reproducible outcomes.

STUDENT RESPONSE AND ADAPTIVE REFINEMENT

Early student response highlighted perceived duplication of communication ("we already discuss work on Messenger"), additional overhead from writing PR descriptions, and delays caused by mandatory peer review. Instructional adjustments focused on behavior rather than tool mastery: review quality was explicitly graded, instructors modeled effective comments, and GitHub onboarding was moved to an earlier course to ensure fluency before advanced project work.

Over successive offerings, PRs became smaller and better scoped, documentation improved, and review discussions became more substantive. This progression is also reflected in rubric-based assessment data from the undergraduate IE course (cf. Table 1), where GitHub practices are introduced. Normalized scores (range $[0, 1]$) across eight team assignments (23 student observations) show a positive trend in both GitHub usage (slope = 0.019, $p = 0.023$) and teamwork and participation (slope = 0.047, $p < 0.001$), indicating increasing adoption of collaborative workflows and more active engagement in shared development practices. These trends are consistent with observed improvements relative to earlier course iterations without structured PR-based collaboration workflows.

ALIGNMENT WITH CDIO STANDARDS

This implementation aligns primarily with CDIO Standards 7, 8, and 11, with partial alignment to Standard 5 (Malmqvist et al., 2020). Integrated Learning Experiences (Standard 7) are realized through PR-centered workflows that integrate technical and interpersonal competencies. Active Learning (Standard 8) emerges through iterative contribution cycles in which students propose changes, receive critique, revise work, and integrate feedback *in situ*. Learning Assessment (Standard 11) is grounded in observable repository artifacts: shared responsibility is

reflected in participation in peer review and approval processes, responsiveness to feedback in explicit replies and revisions within PR discussions, and reproducibility in documentation and execution updates. These behaviors are assessed through rubric-based evaluation of GitHub usage and teamwork. Persistent repositories support aspects of Design-Implement experiences (Standard 5) in semester-long projects.

TRANSFERABLE DESIGN PRINCIPLES

These distilled principles articulate the aspects of the design most likely to transfer to other project-based courses seeking to make collaboration visible and assessable.

1. **Treat collaboration artifacts as assessment evidence.** Use PRs (or equivalent review units) as the basis for grading.
2. **Sequence tool learning before complex project work.** Teach collaboration workflows in a lower-stakes course; assume proficiency later.
3. **Prefer persistent artifacts over fragmented tasks.** Use a single repository for sustained projects to accumulate documentation, decisions, and feedback.
4. **Grade review quality and responsiveness.** Reward substantive critique and explicit responses; penalize cursory approvals or off-platform clarification.
5. **Carry feedback forward.** Treat unresolved comments as work-to-be-done and re-evaluate them at key milestones.

Boundary conditions and limitations. In practice, the approach depends on student readiness for collaborative workflows and familiarity with version control, and introduces overhead from documentation and review requirements, particularly early on. Scalability to larger cohorts requires coordination of review responsibilities, and transferability to non-technical domains depends on the availability of analogous tools supporting transparent iteration and review.

CONCLUSIONS

This paper presented a CDIO implementation in which pull requests (PRs) serve as assessment infrastructure for team-based project courses. By anchoring evaluation in PR authorship and review, explicit responsiveness to feedback, and reproducible handover, the design makes collaboration and individual participation visible within routine project activity. A single persistent repository supports continuity across project cycles and enables efficient capstone re-evaluation through diffs rather than repeated whole-artifact review.

The contribution lies in the assessment architecture rather than in a particular tool: PRs become the unit of feedback and grading, peer review is an assessed responsibility, and unresolved feedback carries forward in a systematic way across the semester. This structure reduces fragmented grading, concentrates attention on reasoning and revision, and embeds documentation and handover as normal habits rather than as end-stage compliance.

Beyond course-level logistics, the approach provides a structural safeguard for individual attainment in team settings. Because PR authorship, critique, and documented follow-up are

attributable to specific students, individual engagement becomes visible even within tightly coupled group work. This helps mitigate free-riding and opaque task ownership and enables instructors to evaluate how each student contributes to the evolving system, not only to isolated deliverables.

At the same time, the workflow aligns closely with professional collaborative practice. PR-based review, documentation, and handover are widely used in industry across software, data science, and technical analysis contexts. Students therefore gain both a fair and transparent assessment structure and sustained practice with a competency directly transferable to modern workplaces. This dual benefit—reliable evidence of individual attainment and authentic preparation for professional collaboration-motivates the design and supports its relevance for CDIO programs.

This work reports practitioner experience rather than controlled experimentation, and its findings should therefore be interpreted as design insights rather than empirical evaluation. Future refinements will explore lightweight review rubrics that calibrate "substantive" critique across teams, clearer PR and issue templates to standardize rationale and handover, and strategies for balancing mixed undergraduate–graduate cohorts.

ACKNOWLEDGEMENTS

The author received no financial support for this work. The author thanks the students whose sustained engagement and willingness to adopt a demanding collaboration and assessment approach made this implementation possible.

REFERENCES

Bennedsen, J., Böjttjer, T., & Tola, D. (2022). Using github classroom in teaching programming. *Proceedings of the 18th International CDIO Conference*.

Biggs, J., & Tang, C. (2015). Constructive alignment: An outcomes-based approach to teaching anatomy. In L. K. Chan & W. Pawlina (Eds.), *Teaching anatomy: A practical guide* (pp. 31–38). Springer International Publishing. https://doi.org/10.1007/978-3-319-08930-0_4

Börjesson, P. O., Hamidian, A., Kubilinskas, E., Richter, U., Weyns, K., & Ödling, P. (2006). Free-riding in group work-mechanisms and countermeasures. *Pedagogiska inspirationskonferensen Genombrottet*, *4*.

GitHub Inc. (2026). Collaborating with pull requests [Accessed: 2026-01-13]. https://docs.github.com/en/pull-requests/collaborating-with-pull-requests

Glassey, R. (2019). Adopting git/github within teaching: A survey of tool support. *Proceedings of the ACM Conference on Global Computing Education*, 143–149. https://doi.org/10.1145/3300115.3309518

Ingimundardóttir, H. (2026). Designing authentic industry-engaged assessment for professional competence in business intelligence. *Proceedings of the 22nd International CDIO Conference*.

Junaid, S., Gorman, P. C., & Leslie, L. J. (2018). Developing logbook keeping as a professional skill through CDIO projects. *Proceedings of the 14th International CDIO Conference*.

Kim, J.-E., & Nembhard, D. A. (2019). The impact of procrastination on engineering students' academic performance. *International Journal of Engineering Education*, *35*(4), 1008.

Malmqvist, J., Edström, K., & Rosén, A. (2020). CDIO standards 3.0—updates to the core CDIO standards. *Proceedings of the 16th International CDIO Conference*, 60–76.

Malmqvist, J., Östlund, S., & Edström, K. (2006). Integrated program descriptions–a tool for communicating goals and design of CDIO programs. *Proceedings of the 2nd International CDIO Conference*. https://cdio.org/knowledge-library/documents/integrated-program-descriptions-tool-communicating-goals-and-design

Petre, M., & Wilson, G. (2014). Code review for and by scientists. In D. Katz (Ed.), *Proceedings of WSSSPE 2014*. https://arxiv.org/pdf/1407.5648

Wilhelm, P., & Nijman, J. (2023). *Academic procrastination in engineering students*. https://doi.org/10.21427/T0KH-SG55

Wilson, G., Bryan, J., Cranston, K., Kitzes, J., Nederbragt, L., & Teal, T. K. (2017). Good enough practices in scientific computing. *PLOS Computational Biology*, *13*(6), e1005510. https://doi.org/10.1371/journal.pcbi.1005510

BIOGRAPHICAL INFORMATION

Helga Ingimundardóttir is an Assistant Professor of Industrial Engineering at the University of Iceland. Since 2023, her teaching and research have focused on optimization, mathematical modeling, data science, artificial intelligence, and industry-engaged learning in data-intensive contexts. She is a member of the Teaching Academy of the public universities in Iceland (2024). Helga brings extensive industry experience across research and applied AI: as a Research Scientist at *deCODE Genetics*, she designed long-read sequencing analysis pipelines and co-authored papers in *Nature Genetics* and *Genome Biology*; as a Data Scientist at *CCP Games*, she helped develop real-time recommendation services for *EVE Online*; as Head of AI Research at *Travelshift*, she led large-scale trip-planning optimization; and earlier, as a Computational Engineer in R&D at *Valka* (now JBT Marel), she developed X-ray-based optimization methods for automated fish processing. She holds a PhD and MSc in Computational Engineering and a BSc in Mathematics from the University of Iceland, as well as a graduate diploma in Teaching Studies for Higher Education.

Corresponding author

Helga Ingimundardóttir
University of Iceland
School of Engineering and Natural Sciences
Faculty of Industrial Engineering, Mechanical Engineering and Computer Science
VR-II, Hjarðarhagi 6
107 Reykjavík, ICELAND
helgaingim@hi.is